MC-MXI
User Manual

May 1994 Edition

Part Number 320297-01

© Copyright 1994 National Instruments Corporation.
All Rights Reserved.

National Instruments Corporate Headquarters
6504 Bridge Point Parkway
Austin, TX 78730-5039
(512) 794-0100
Technical support fax: (800) 328-2203
(512) 794-5678

Branch Offices:
Australia (03) 879 9422, Austria (0662) 435986, Belgium 02/757.00.20, Canada (Ontario) (519) 622-9310,
Canada (Québec) (514) 694-8521, Denmark 45 76 26 00, Finland (90) 527 2321, France (1) 48 14 24 24,
Germany 089/741 31 30, Italy 02/48301892, Japan (03) 3788-1921, Netherlands 03480-33466, Norway 32-848400,
Spain (91) 640 0085, Sweden 08-730 49 70, Switzerland 056/20 51 51, U.K. 0635 523545

Limited Warranty

The MC-MXI is warranted against defects in materials and workmanship for a period of one year from the date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Under the copyright laws, this book may not be copied, photocopied, reproduced, or translated, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

LabVIEW® and NI-VXI™ are trademarks of National Instruments Corporation.

Product names listed are trademarks of their respective manufacturers. Company names listed are trademarks or trade names of their respective companies.

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

FCC/DOC Radio Frequency Interference Compliance

This equipment generates and uses radio frequency energy and, if not installed and used in strict accordance with the instructions in this manual, may cause interference to radio and television reception. This equipment has been tested and found to comply with the following two regulatory agencies:

Federal Communications Commission

This device complies with Part 15 of the Federal Communications Commission (FCC) Rules for a Class A digital device. Operation is subject to the following two conditions:

1. This device may not cause harmful interference in commercial environments.

2. This device must accept any interference received, including interference that may cause undesired operation.

Canadian Department of Communications

This device complies with the limits for radio noise emissions from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications (DOC).

Le présent appareil numérique n'émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de classe A prescrites dans le règlement sur le brouillage radioélectrique édicté par le ministère des communications du Canada.

Instructions to Users

These regulations are designed to provide reasonable protection against harmful interference from the equipment to radio reception in commercial areas. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

There is no guarantee that interference will not occur in a particular installation. However, the chances of interference are much less if the equipment is installed and used according to this instruction manual.

If the equipment does cause interference to radio or television reception, which can be determined by turning the equipment on and off, one or more of the following suggestions may reduce or eliminate the problem.

- Operate the equipment and the receiver on different branches of your AC electrical system.
- Move the equipment away from the receiver with which it is interfering.
- Reorient or relocate the receiver's antenna.
- Be sure that the equipment is plugged into a grounded outlet and that the grounding has not been defeated with a cheater plug.

Notice to user: Changes or modifications not expressly approved by National Instruments could void the user's authority to operate the equipment under the FCC Rules.

If necessary, consult National Instruments or an experienced radio/television technician for additional suggestions. The following booklet prepared by the FCC may also be helpful: *How to Identify and Resolve Radio-TV Interference Problems.* This booklet is available from the U.S. Government Printing Office, Washington, DC 20402, Stock Number 004-000-00345-4.

Contents

About This Manual xi
Organization of This Manual xi
Conventions Used in This Manual xii
How to Use This Manual xii
Related Documentation xii
Customer Communication xii

Chapter 1
Introduction to MXIbus 1-1
Overview 1-1
The Need for MXIbus 1-1
MXIbus Applications 1-2
MXIbus Operation 1-4
MXIbus Device Functions 1-4
MXIbus Cables 1-5
MXIbus Termination 1-6
MXIbus Signals 1-6
MXIbus Byte Ordering 1-8
Bus Arbitration 1-9
Addressing Modes and Address Spaces 1-10
Data Transfers 1-12
Basic Data Transfers 1-12
Block-Mode Data Transfers 1-13
Indivisible Data Transfers 1-13
Priority Select Data Transfers 1-14
Interrupts 1-14
Relationship between the MXIbus, VXIbus, and MC-MXI 1-14
Configuration Registers 1-15
VXIbus Device Classes 1-15
Word Serial Protocol 1-16
MXIbus Performance 1-16
Comparing Data Transfer Rates 1-16
Local Performance 1-18

Chapter 2
General Information 2-1
Overview 2-1
What Your Kit Should Contain 2-3
Optional Equipment 2-3
Optional Software 2-4

Chapter 3
Configuration and Installation 3-1
Unpacking 3-1
Configuration 3-1
Installation 3-2

Chapter 4
Register Descriptions

Register Descriptions 4-1
Overview 4-1
Cross-Referencing Registers 4-1
Definitions 4-1
Register Bit Fields with Constant Values 4-2
Register Bit Fields with Undefined Value 4-2
Read-Only and Write-Only Register Bit Fields 4-2
Scope of Resets 4-2
Register Byte Ordering 4-2
Register Map 4-2
Basic MXIbus Registers 4-6
MXI ID Register 4-6
MXI Logical Address Register 4-8
MXI Device Type Register 4-9
MXI Status Register 4-10
MXI Control Register 4-12
MXI Offset Register 4-13
Message-Based Device Registers 4-14
MXI Protocol Register 4-14
MXI Signal Register 4-16
MXI Response Register 4-18
Data Registers 4-22
Data Extended (In) 4-22
Data High (In) 4-23
Data Low (In) 4-23
Data High (Out) 4-24
Data Low (Out) 4-24
A24 Pointer Register 4-25
A24 Pointer (High) 4-25
A24 Pointer (Low) 4-26
A32 Pointer Register 4-27
A32 Pointer (High) 4-27
A32 Pointer (Low) 4-27
Soft Reset Service Register 4-28
Communication Status Register 4-29
Communication Control Register 4-31
Slave Configuration Register 4-33
MC-MXI Local Registers 4-35
PITLOCAL Register 4-35
PITREADY Register 4-37
PITSC Register 4-39
PITCNTR Register 4-41
WIN1AM-SMOFF Register 4-43
WIN3AM-WIN2AM Register 4-45
Master Mode Page Register 4-46
MC-MXI–STATUS Register 4-47
MC-MXI–CTRL Register 4-50
WAITSTATES Register 4-53
DMA Enable Register 4-55

Interrupt Enable Register 4-56
MC INT Status Register 4-58
MC INT Ack. Register 4-60
SCCLK Register 4-61
DMA Control Register 4-62
POS Registers 4-64
POS 0/1 MC-MXI Micro Channel ID Register 4-64
POS 2 MC Setup/DMA Channel 1 Register 4-65
POS 3 Memory/IO Base Address Register 4-67
POS 4 MC Interrupt Map Register 4-69
POS 5 MC Arbitration/DMA Channel 2 Register 4-71

Chapter 5
Programming the MC-MXI 5-1
Initializing the MC-MXI 5-1
Step 1. Set Up the MC-MXI POS Registers 5-2
Step 2. Specify MC-MXI Wait States 5-3
Step 3. Specify the Cycle Period of the System Controller Base Clock 5-3
Step 4. Initialize the Timers 5-4
Step 5. Enable the MXIbus System Controller 5-5
Operating the MC-MXI 5-5
MXIbus Master-Mode Operation 5-6
Operating the MC-MXI under a DOS Environment 5-7
Accessing MXIbus Memory in Master Mode 5-7
Memory Paging 5-7
Deadlock 5-9
Timing Incompatibilities 5-10
Programming the MC-MXI for Block-Mode Transfers 5-10
MXIbus Slave-Mode Operation 5-12
Controlling Interrupts between the MXIbus and the Micro Channel 5-14
Recognizing COMM Interrupts 5-15
Recognizing MISC Interrupts 5-16
Recognizing MMERR Interrupts 5-17
Recognizing MXIRQ Interrupts 5-17
Generating and Handling Interrupts 5-17
Byte Swapping 5-19

Chapter 6
Theory of Operation 6-1
MC-MXI Architecture 6-1
MXIbus Terminators 6-3
Slave-Mode Address Decoder 6-3
Slave-Mode Address Formation 6-3
Slave-Mode State Machine 6-3
Master-Mode Address Decoder 6-3
Master-Mode Address Formation 6-4
Master-Mode State Machine 6-4
Master-Mode Address Modifiers 6-4
Parity Generator/Checker 6-4
Byte-Swapping Logic 6-4

Interrupt Generation and Recognition....6-4
VXIbus Device Support....6-5
Address, Data, and Control Transceivers....6-5
Master-Mode Operation....6-5
Enabling Master-Mode Operation....6-6
MXIbus Arbitration....6-6
MXIbus Address Broadcast....6-6
Master-Mode Data Transfer....6-7
Master-Mode Cycle Termination....6-7
Data Transfers Using Indivisible Operations....6-8
Master-Mode Block Transfers....6-9
Slave-Mode Operation....6-9
Enabling Slave-Mode Operation....6-9
MC-MXI Slave-Mode Address Mapping....6-10
Mapping A16 Space to the MC-MXI Register Set....6-10
Micro Channel Bus Arbitration....6-11
Slave-Mode Data Transfer....6-11
Slave-Mode Cycle Termination....6-11
Slave-Mode Block Transfers....6-12
Reset Circuitry on the MC-MXI....6-13

Appendix A
Specifications....A-1

Appendix B
Mnemonics Key....B-1

Appendix C
MXIbus Connector Description....C-1

Appendix D
Customer Communication....D-1

Glossary....Glossary-1

Index....Index-1

Figures

Figure 1-1. PC Using MXI to Control VXIbus or VMEbus 1-2
Figure 1-2. MXI Used for Multiple Mainframe VXIbus or VMEbus System 1-3
Figure 1-3. MXI Used for High-Speed Shared-Memory Network 1-3
Figure 1-4. MXIbus Multi-Drop Cable Assembly 1-5
Figure 1-5. Typical MXIbus Application 1-17

Figure 2-1. MC-MXI Interface Board 2-1

Figure 3-1. MXIbus Terminating Networks 3-2

Figure 5-1. Logic to Recognize COMM Interrupts 5-15
Figure 5-2. Logic to Recognize MISC, MXIRQ, and MMERR Interrupts 5-16
Figure 5-3. Logic to Generate Micro Channel Interrupts 5-18

Figure 6-1. MC-MXI Architecture 6-2

Figure C-1. MXIbus Connector C-1

Tables

Table 1-1. MXIbus Signal Meanings 1-7
Table 1-2. Data Transfer Types 1-8
Table 1-3. Address Modifier Codes 1-10

Table 4-1. MC-MXI Register Map 4-4
Table 4-2. POS Register Map 4-5

Table 5-1. Pseudo-Code Instructions 5-1
Table 5-2. Recommended Count Values for MC-MXI Timers 5-5
Table 5-3. MXIbus Address Formation Using the Master Mode PAGE Register 5-8
Table 5-4. Common Byte-Ordering Schemes 5-19
Table 5-5. Byte-Swapping Operations as Function of Data Transfer Size 5-20

Table 6-1. Micro Channel Memory Addressing Modes 6-10

Table C-1. MXIbus Connector Signal Assignments C-1
Table C-2. MXIbus Signal Groupings C-2

About This Manual

The MC-MXI is an interface. It links computer systems that are based on the Micro Channel to the MXIbus. This manual examines the MC-MXI at a technical level. This manual is particularly suited to software and hardware engineers who are integrating the MC-MXI into a larger system and who have elected not to use the standard NI-VXI bus interface software provided by National Instruments. To better understand this manual, you should be familiar with the Micro Channel bus architecture and the operations it supports.

Organization of This Manual

The *MC-MXI User Manual* is organized as follows:

- Chapter 1, *Introduction to MXIbus*, provides an overview of the MXIbus.
- Chapter 2, *General Information*, contains an overview of the functionality of the MC-MXI interface board, shows a picture of the MC-MXI board, and lists the contents of your kit and the available optional equipment.
- Chapter 3, *Configuration and Installation*, describes the procedures for unpacking, configuring, and installing your MC-MXI interface board.
- Chapter 4, *Register Descriptions*, contains information on the use of the MC-MXI registers.
- Chapter 5, *Programming the MC-MXI*, discusses important initialization and normal-operation programming aspects of the MC-MXI.
- Chapter 6, *Theory of Operation*, contains a functional block diagram of the MC-MXI, a brief description of the major elements of the interface board, and a detailed description of both master-mode and slave-mode operation.
- Appendix A, *Specifications*, lists various MC-MXI module specifications, such as physical dimensions and power requirements.
- Appendix B, *Mnemonics Key*, contains an alphabetical listing of the mnemonics that this manual uses to describe signals, registers, and register bits.
- Appendix C, *MXIbus Connector Description*, describes the connector pin assignments for the MXIbus connector.
- Appendix D, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.
- The *Glossary* contains an alphabetical list and a description of the terms that this manual uses, including abbreviations, acronyms, metric prefixes, and symbols.
- The *Index* contains an alphabetical list of the key terms and topics that this manual uses, and it includes the page number where you can find each term and topic.

Conventions Used in This Manual

The following conventions are used in this manual:

italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept. In this manual, italic text also denotes signal names.
`monospace`	Lowercase text in this font denotes text or characters that you literally input from the keyboard. This font also denotes sections of code and programming examples.

Numbers in this manual are assumed to be decimal unless followed by an *h* suffix, which denotes hexadecimal numbers (for example, D5h).

Abbreviations, acronyms, metric prefixes, mnemonics, symbols, and terms are listed in the *Glossary*.

How to Use This Manual

You should begin with Chapter 1 to gain an understanding of MXIbus concepts. This chapter explains how MXIbus devices attach together and communicate with each other. Chapter 2 contains a general overview about the MC-MXI board. Chapter 3 contains information on how to configure and install your MC-MXI into a Micro Channel computer. Chapters 4 and 5 contain information you will need when programming your MC-MXI: you can skip these chapters if you are using a compatible National Instruments software package, because the software routines perform these functions automatically. Chapter 6 contains more technical information on the use of the MC-MXI.

Related Documentation

The following manuals contain information that you may find helpful as you read the *MC-MXI User Manual*:

- *Multisystem Extension Interface Bus Specification*, Version 1.2 (part number 340007-01)
- VXI-1, *VXIbus System Specification*, Revision 1.4, VXIbus Consortium (available from National Instruments, part number 350083-01)

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and configuration forms for you to complete. These forms are in Appendix D, *Customer Communication*, at the end of this manual.

Chapter 1
Introduction to MXIbus

This chapter provides an overview of the MXIbus; it begins by briefly discussing the motivations behind the MXIbus standard, then it presents potential MXIbus applications. The remainder of this chapter presents a more technical discussion of the important elements of the MXIbus standard: you need to know these elements to better understand the MC-MXI.

Overview

The MXIbus (Multisystem Extension Interface Bus) is a high-performance communication link that interconnects devices by using round, flexible cables. MXIbus operates like modern backplane computer buses, but it is a cabled communication link for high-speed communication between physically separate devices. The emergence of the VXIbus inspired the MXIbus. National Instruments, a member of the VXIbus Consortium, recognized that VXI requires a new generation of connectivity for the instrumentation systems of the future. National Instruments developed the MXIbus specification over a period of two years and announced it in April 1989 as an open industry standard.

You can use MXIbus interface products in a variety of platforms, including the VXIbus and VMEbus backplane systems, and the IBM PC AT, EISA, PS/2, Sun SPARCstation, DECstation 5000, RISC System/6000, and Macintosh computer systems. MXIbus products directly and transparently couple these industry-standard computers to the VXIbus and the VMEbus backplanes. They also transparently extend VXI/VME across multiple mainframes, and they seamlessly integrate external devices that cannot physically fit on a plug-in module into a VXI/VME system.

The Need for MXIbus

Sophisticated I/O architectures can now move data at rates exceeding 10 MB/s. At the same time, modern peripherals, such as color scanners and printers, and instruments, such as digitizers, logic analyzers, and digital test subsystems, generate vast amounts of data at ever-increasing data rates. The capabilities of MXIbus have become increasingly useful for applications that use these data-intensive peripherals.

Clearly, the I/O capabilities of modern PCs and workstations can handle data-intensive instrumentation applications. However, the industry has lacked a standard communication link that interconnects devices so that they can operate at full speed across the connection. The worldwide GPIB standard, which was initially designed in the mid 1960s, is 30 years old and relatively slow. Some of the latest networks have higher burst data rates than GPIB, but are not appropriate for real-time, data-intensive applications because their heavy protocol overhead is geared for efficient passing of small message packets.

A memory-mapped communication system that transparently extends bus-level I/O transactions between systems is an ideal solution. It completely eliminates software protocol overhead, provides direct control and shared memory between devices, and matches the data rates of high-performance computers and peripherals. The MXIbus is such a communication system.

MXIbus Applications

A computer, instrument, or other device with a MXI interface is called a MXIbus device. Typically, MXIbus devices are systems or instruments that have a MXI interface board installed. Most MXIbus devices have their own internal system bus for internal communication. The MXI interface board connects this internal bus to the MXIbus.

Figures 1-1, 1-2, and 1-3 show three examples of MXIbus applications. MXIbus gives external computers direct control of the VXIbus, so it is as if they were embedded in the VXI mainframe. A VXI-to-MXI mainframe extender can extend VXI to multiple mainframes. Software is also available for VXI programming.

VMEbus systems are another target application for MXIbus products. You can use VME interface kits to directly control the VMEbus, and you can use a VME-to-MXI chassis extender to extend VME for multiple-chassis configurations. Software is available for VME programming.

In addition to VXI and VME applications, you can use MXIbus interface products in a variety of general-purpose, computer-to-computer applications. You can mix and match MXIbus products to interconnect any number of MXIbus devices for high-performance communication. For example, MXIbus can connect PC AT, EISA, PS/2, SunSPARCstation, DECsystem 5000, RISC System/6000, Macintosh, and other computers and workstations for a high-speed, shared-memory network. You can order MXIbus computer interfaces individually.

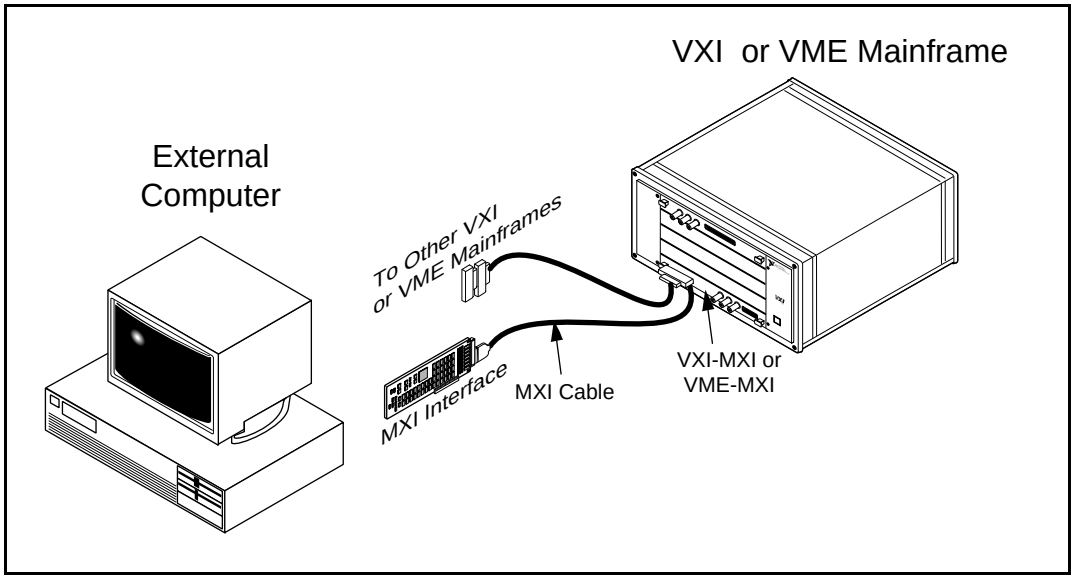


Figure 1-1. PC Using MXI to Control VXIbus or VMEbus

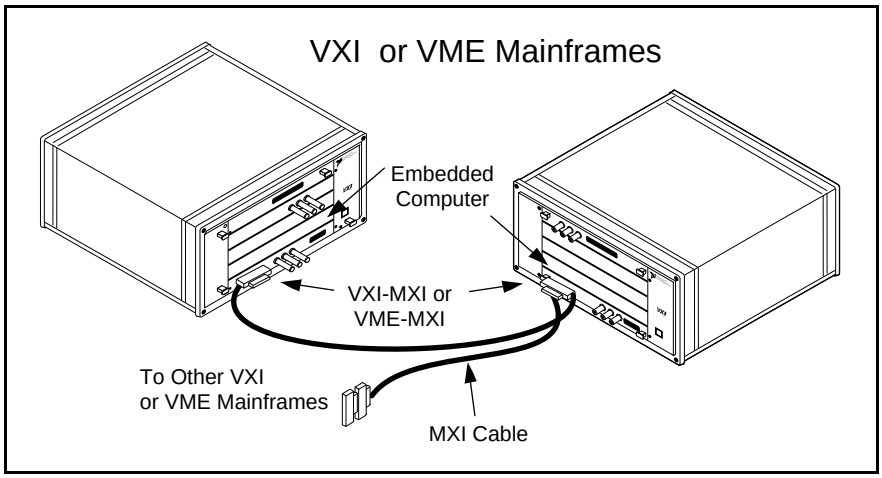


Figure 1-2. MXI Used for Multiple Mainframe VXIbus or VMEbus System

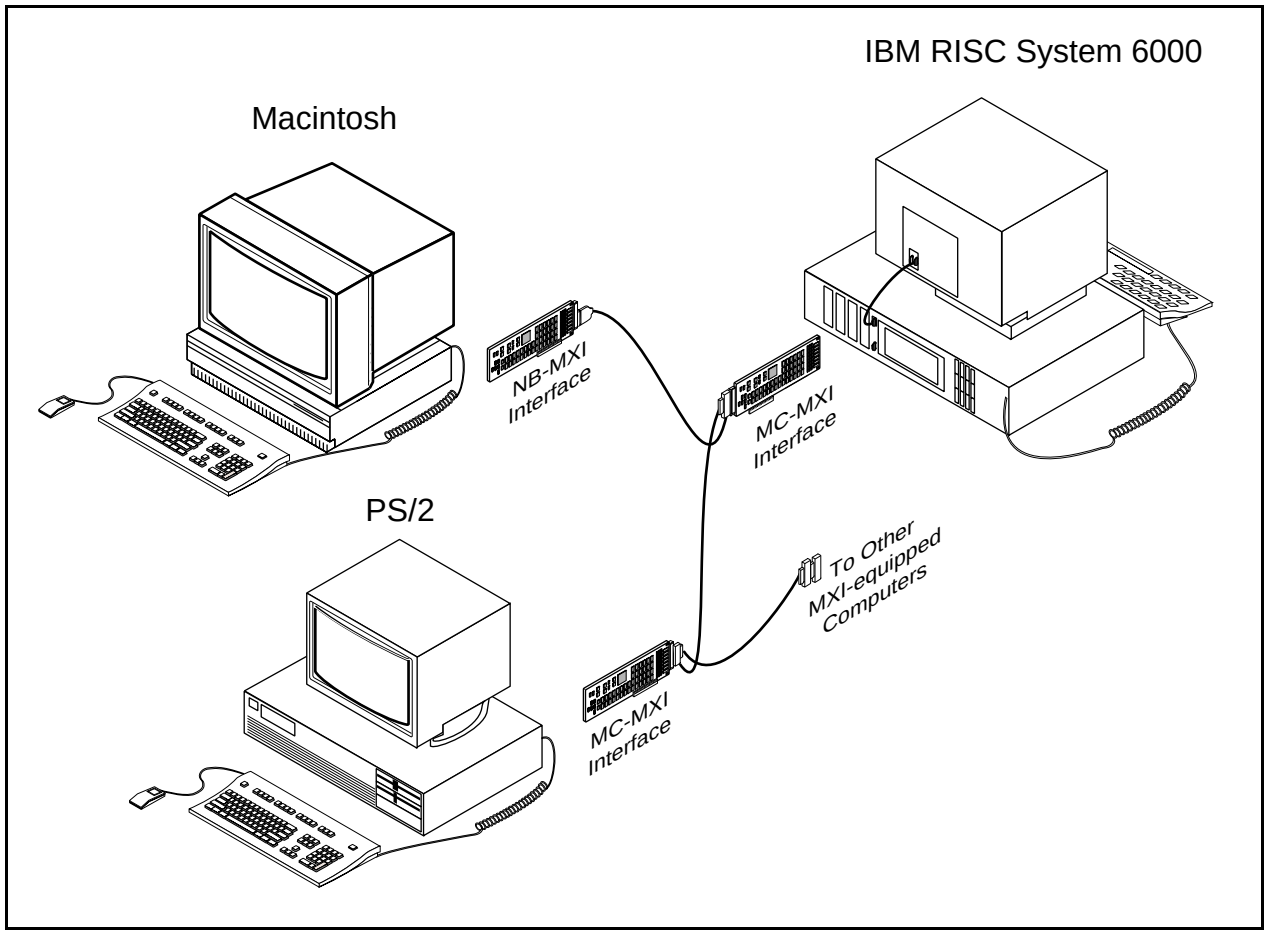


Figure 1-3. MXI Used for High-Speed Shared-Memory Network

MXIbus Operation

MXIbus is a general-purpose, 32-bit, multimaster system bus on a cable. MXIbus uses a hardware memory-mapped communication scheme that reduces software overhead. You can daisy-chain together up to eight MXIbus devices.

MXIbus tightly couples multiple devices by mapping together portions of their individual memory address spaces. MXIbus devices accomplish this mapping by interconnecting at the hardware level and operating as a single system with a shared address space. MXIbus devices can directly access each other's resources by performing simple reads and writes to appropriate address locations.

To help you understand how the MC-MXI operates, it is important for you to know some fundamental MXIbus concepts. This discussion describes the basic features of the MXIbus and how these features combine to transfer information. These features are as follows:

- MXIbus device functions
- MXIbus cables and termination
- MXIbus signals
- Byte ordering on the MXIbus
- Bus arbitration
- Addressing modes and address spaces
- Data transfers
- Interrupts
- Relationship between the MXIbus and VXIbus
- Performance considerations

For a more thorough description of the MXIbus, consult the *MXIbus Specification* (National Instruments part number 340007-01).

MXIbus Device Functions

MXIbus devices can perform three different functions: *master*, *slave*, and *System Controller*. These functions have specific meanings and are defined as follows:

- A *MXIbus master* initiates bus cycles to transfer data between itself and a MXIbus slave device.
- A *MXIbus slave* detects bus cycles initiated by the MXIbus master and, when those cycles select the slave, transfers data between itself and the MXIbus master.

- A *MXIbus System Controller*, which is a functional module on a MXIbus device that performs arbitration, generates MXIbus cycle timeouts. A MXIbus network has only one active System Controller, which is always the first device in the MXIbus daisy-chain.

MXIbus devices can perform master functions, slave functions, or both. A MXIbus device can also implement the System Controller function as long as this function is independent of its requirements as a master and/or slave.

MXIbus Cables

There are two basic types of MXIbus cables. One type of MXIbus cable is a *point-to-point cable* with a single connector on each end. The other type of MXIbus cable is known as a *multi-drop cable*; it has a single connector on one cable end and a double connector on the other end. A MXIbus system consists of two or more MXIbus devices connected in a daisy-chain fashion. Every MXIbus system has one MXIbus device that acts as the MXIbus System Controller. The MXIbus System Controller must be the first device in the daisy-chain. Therefore, the System Controller must have a single-connector cable end. Subsequent devices will have the double connector end.

Figure 1-4 is a diagram of the multi-drop type of cable assembly used in a daisy-chained MXIbus system. You can daisy-chain additional devices to the double connector to propagate the bus. You should use a MXIbus cable with a single connector on each end when the system contains only two MXIbus devices or when you are connecting the last cable section in the daisy-chain.

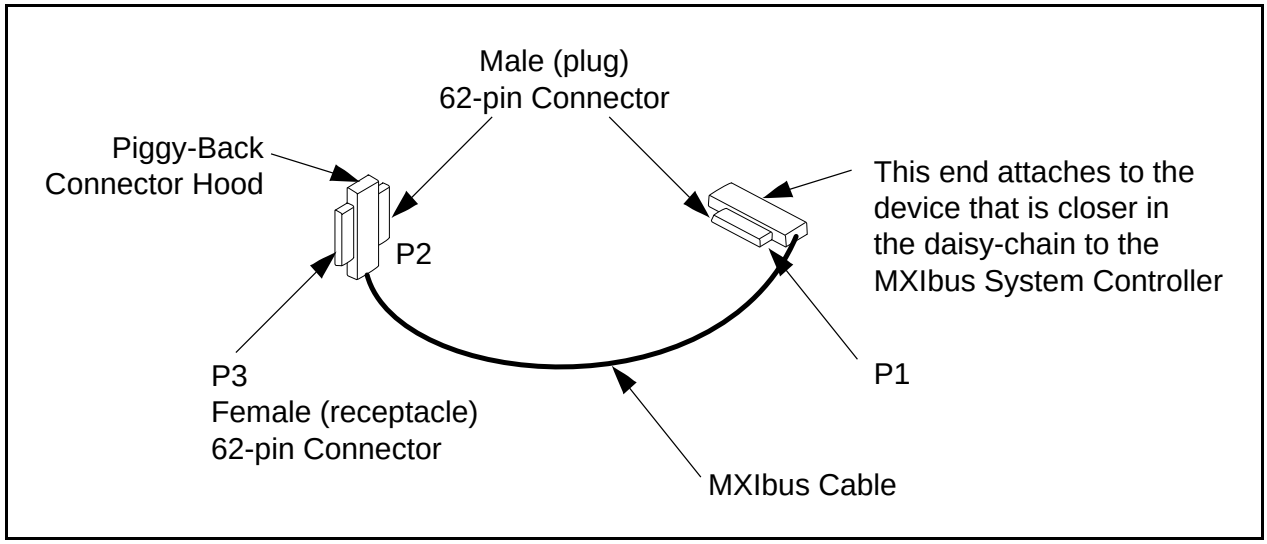


Figure 1-4. MXIbus Multi-Drop Cable Assembly

A single MXIbus cable can be any length up to 20 m. If you are daisy-chaining multiple MXIbus devices together, the total cable distance must be no more than 20 m. The MXIbus cable is a flexible, round cable similar to a GPIB cable (about 0.6 in. in diameter). Internally there are 48 single-ended, twisted-pair signal lines. Double shielding with an aluminum mylar shield as well as a copper braid shield eliminates any EMI problems. The stacking depth of two daisy-chained MXIbus cables is approximately 3.4 in.

Each MXIbus signal line is twisted with its own ground line. All MXIbus signal lines have matched impedance to minimize signal skew and reflections. Limiting stub lengths to no more than 4 in. off the mainline interconnection minimizes reflections because of impedance discontinuities. You must have termination networks at the first and last MXIbus devices to minimize reflections at the ends of cables.

MXIbus uses state-of-the-art, single-ended, trapezoidal bus transceivers to reduce noise crosstalk in the transmission system. Designed specifically for driving backplane bus signals, these transceivers have open-collector drivers that generate precise trapezoidal waveforms with typical rise and fall times of 9 ns. The trapezoidal shape, which is the result of constant rise and fall times, reduces noise coupling (crosstalk) on adjacent lines. The receiver uses a lowpass filter to remove noise and a high-speed comparator that recognizes the trapezoidal-shaped signal from the noise.

MXIbus Termination

The MXIbus requires that the first and last devices in the daisy-chain have a termination network. Two basic types of termination networks are available. Some MXIbus devices have onboard termination schemes that should be enabled on the end devices of the daisy-chain. You can also use external terminating packs for easy system reconfiguration and for MXIbus devices that lack onboard terminating networks. MXIbus devices other than the two end devices should *not* have an external terminating pack and must have any onboard terminating networks defeated. Also, each end device must have only one of these termination options.

MXIbus Signals

The MXIbus connector is a single, rugged, high-density, 62-pin, D-subminiature connector. MXIbus signals include 32 multiplexed address and data lines with parity, address modifiers for multiple address spaces, single-level multimaster prioritized bus arbitration, and several control signals. You can perform 32-bit, 16-bit, and 8-bit data transfers, as well as indivisible read/write operations and integrated block-mode transfers. The maximum data rate for MXIbus is 20 MB/s.

Table 1-1 explains the meaning of each group of signals. Refer also to Appendix C, *MXIbus Connector Description*, to see the connector pin assignments on the MXIbus connector.

Table 1-1. MXIbus Signal Meanings

Signal	Mnemonic	Description
Address Modifier	*AM[4–0]**	These signals specify the address space and the type of data transfer.
Address/Data	*AD[31–0]**	These signals comprise a 32-bit-wide path for address and data.
Address Strobe	*AS**	A MXIbus master asserts this signal to indicate that it has placed a valid address on the MXIbus.
Data Strobe	*DS**	A MXIbus master asserts this signal to indicate that it has placed valid data on the MXIbus or to request data from a slave.
Write/Read	*WR**	When asserted low by a MXIbus master, this signal indicates that the current data transfer is a write operation. When asserted high, this signal indicates that the current data transfer is a read operation.
Data Transfer Acknowledge	*DTACK**	MXIbus slaves assert *DTACK** to indicate successful completion of a data transfer.
Bus Error	*BERR**	MXIbus masters and slaves assert this signal to indicate an error condition.
Parity	*PAR**	Denotes the parity of the *AD[31–0]** signals. *PAR** is asserted either high or low so that combining *PAR** with *AD[31–0]** results in an even number of true (low) bits. Parity is always calculated over all the bits in the *AD** path.
Transfer Size	*SIZE**	Used in conjunction with the *AD0** and *AD1** signals (when these signals are transmitting address information) to denote the width and byte ordering of the data path.
Bus Request	*BREQ**	MXIbus masters assert this signal to request access to the MXIbus.
Bus Busy	*BUSY**	This signal indicates that the MXIbus is in use.
Bus Grant In	*GIN**	This signal informs a MXIbus device that it owns the MXIbus.
Interrupt Request	*IRQ**	This signal is asserted to initiate a MXIbus interrupt.
Terminator Power	*TERMPWR*	Supplies 3.4 V to pull up resistors on the MXIbus.
Signal Ground	*GND*	Electrical ground.

MXIbus Byte Ordering

The MXIbus has a 32-bit-wide data path and uses the same byte-ordering scheme as the VXIbus and the VMEbus to transfer information. The MXIbus supports 32-bit, 16-bit, and 8-bit data transfers.

Table 1-2 shows the order in which bytes are transmitted on the *AD** signals as a function of *SIZE**, *AD0**, and *AD1**. To better understand byte ordering, consider that the 32-bit MXIbus data bus is divided into four separate byte lanes (*AD[31–24]**, *AD[23–16]**, *AD[15–8]**, and *AD[7-0]**). The logical state of the *SIZE**, *AD1**, and *AD0** signals (H = logic high; L = logic low) determines the size of the data transfer and the ordering of the data bytes on the MXIbus data path.

During a 32-bit data transfer, four bytes of data are transmitted over the four byte lanes as indicated in Table 1-2. These bytes (Byte[0]–Byte[3]) come from four consecutive memory locations in the MXIbus device with Byte[0] at the lowest address and Byte[3] at the highest address.

If a MXIbus device attempts to transmit the same four bytes of data by using a 16-bit or 8-bit data transfer, the bytes do not map directly to the MXIbus byte lanes as they do with 32-bit data transfers. In addition, 16-bit and 8-bit data transfers require two or four data transfers, respectively, to move all four bytes across the MXIbus. The MXIbus does impose a specific byte ordering on data transfers. The *Byte Swapping* section of Chapter 5, *Programming the MC-MXI*, explains in greater detail how the MC-MXI handles byte ordering between the Micro Channel and the MXIbus.

Table 1-2. Data Transfer Types

Transfer Size	***SIZE****	***AD1****	***AD0****	***AD31*–AD24****	***AD23*–AD16****	***AD15*–AD08****	***AD07*–AD00****
8-bit (Byte) (4 transfers)	H H H H	H H L L	H L H L			Byte(0) Byte(2)	 Byte(1) Byte(3)
16-bit (Word) (2 transfers)	L L	H L	H H			Byte(0) Byte(2)	Byte(1) Byte(3)
32-bit	L	L	L	Byte(0)	Byte(1)	Byte(2)	Byte(3)
Reserved	L	H	L	Undefined	Undefined	Undefined	Undefined

Bus Arbitration

The MXIbus provides arbitration capabilities among multiple master devices. The arbitration scheme is based on the physical location of the MXIbus devices on the bus. For each device, the *GOUT** signal of one device is connected to the *GIN** signal of the next device. The MXIbus System Controller is always the first device on the MXIbus. The daisy-chain connection of *GOUT** and *GIN** imposes a hierarchy on the devices on the MXIbus. A device that is physically located before another device on the daisy-chain has a higher priority.

If a MXIbus master requires access to the MXIbus, it asserts *BREQ**. The System Controller detects the bus request and initiates bus arbitration when the *BUSY** signal is unasserted (that is, the current data transaction is completed).

The System Controller asserts the *GOUT** signal, which passes to the *GIN** signal of the next device on the MXIbus. If a device receives *GIN** while it is asserting *BREQ**, it unasserts *BREQ**, asserts *BUSY**, and begins its data transfer. If the device is not asserting *BREQ**, it passes the *GIN** signal to the *GOUT** signal of the device.

*BREQ** can be driven simultaneously by multiple MXIbus devices. If a device has completed its transfer and *BREQ** is still asserted, the System Controller initiates arbitration once again.

You can configure MXIbus devices to use a fairness mode. When fairness is implemented, a MXIbus device that has won arbitration for the bus does not assert *BREQ** again until it detects that *BREQ** is unasserted. By contrast, a MXIbus device can *lock* the MXIbus by asserting *BUSY** for as long as the device requires the bus.

Arbitration serves a dual purpose on the MXIbus. It is a means for MXIbus master devices to access the bus in an orderly manner. Arbitration also is a means for slaves mapped to the same address space to determine which slave will participate in a data transfer. This latter purpose is called *priority select* and is described in the *Data Transfers* section in this chapter.

Addressing Modes and Address Spaces

A MXIbus device has three different address spaces and different methods of accessing them. The address modifier signals (*AM[4–0]**) reference these different address spaces, define access privilege, and determine the type of data transfer. Table 1-3 defines the meaning of the different address modes.

Table 1-3. Address Modifier Codes

Address Modifier Line					Function
AM4*	**AM3***	**AM2***	**AM1***	**AM0***	
L	L	L	L	L	A24, supervisory, block transfer
L	L	L	L	H	A24, supervisory, program access
L	L	L	H	L	A24, supervisory, data access
L	L	L	H	H	Reserved
L	L	H	L	L	A24, nonprivileged, block transfer
L	L	H	L	H	A24, nonprivileged, program access
L	L	H	H	L	A24, nonprivileged, data access
L	L	H	H	H	Reserved
L	H	L	L	L	Reserved
L	H	L	L	H	Reserved
L	H	L	H	L	A16, supervisory access
L	H	L	H	H	Reserved
L	H	H	L	L	Reserved
L	H	H	L	H	Priority selection cycle
L	H	H	H	L	A16, nonprivileged access
L	H	H	H	H	Reserved
H	L	L	L	L	User defined
H	L	L	L	H	User defined
H	L	L	H	L	User defined
H	L	L	H	H	User defined
H	L	H	L	L	User defined
H	L	H	L	H	User defined
H	L	H	H	L	User defined
H	L	H	H	H	User defined
H	H	L	L	L	A32, supervisory, block transfer
H	H	L	L	H	A32, supervisory, program access
H	H	L	H	L	A32, supervisory, data access
H	H	L	H	H	Reserved
H	H	H	L	L	A32, nonprivileged, block transfer
H	H	H	L	H	A32, nonprivileged, program access
H	H	H	H	L	A32, nonprivileged, data access
H	H	H	H	H	Reserved

MXIbus devices access each other through three different address spaces. If a MXIbus device is capable of both master and slave operations, the device can map into one address space for master-mode operations and map to a different address space for slave-mode operations. The three address spaces are as follows:

- *A16 space* is 64 KB *in size* and is accessible by using the 16 least significant address bits (*AD[15–0]**) on the MXIbus.
- *A24 space* is 16 MB *in size* and is accessible by using the 24 least significant address bits (*AD[23–0]**) on the MXIbus.
- *A32 space* is 4 GB *in size* and is accessible by using all of the 32 address bits (*AD[31–0]**) on the MXIbus.

A16, A24, and A32 spaces are completely independent of each other. For example, A16 is *not* the lowest 64 KB of A32 or A24 space.

Each address space is accessible via different privileges and modes. A24 and A32 spaces are accessible through the following four types of privileges:

- Supervisory data access
- Supervisory program access
- Nonprivileged data access
- Nonprivileged program access

A16 space is accessible through the following two types of privileges:

- Supervisory access
- Nonprivileged access

The implementation of these privileges on MXIbus devices is flexible. A MXIbus device can ignore these privileges, or it can use the privileges to permit only certain types of accesses to and from the device.

The address modifiers also determine the types of data transfers that can take place on the MXIbus. Data transfers are explained in the following section, but for now it is important to note that address modifiers play a key role in the process.

Data Transfers

Up to this point, the discussion of the MXIbus has focused on its individual features and components. These features and components work together to perform data transfers. The MXIbus supports four types of data transfers:

- Basic
- Block
- Indivisible
- Priority Select

Basic Data Transfers

The most fundamental data transfer is the basic transfer involving the exchange of a single data word between a master and a slave. A word of data can be one to four bytes, depending on the state of the *SIZE**, *AD0**, and *AD1** signals. The steps involved in performing a basic data transfer are as follows:

1. The master device initiates the transfer by driving *AD[31–0]**, *PAR**, *WR**, *SIZE**, and *AM[4-0]** onto the MXIbus. *AD[31–0]** carries the desired address for the data transfer. A MXIbus slave existing at that address recognizes that a data transfer is about to take place.

2. The master asserts *AS** to inform the slave that the address information on the MXIbus is valid. The slave then latches the address information.

3. During a write operation, the master places data onto the MXIbus and asserts *DS**. The slave then latches the data. During a read operation, the master asserts *DS** to instruct the slave to place data onto the MXIbus. At some point during this part of the data transfer, the master unasserts *AS**.

4. Once the slave has either received the data from the master or placed data on the bus (for read and write operations, respectively), it asserts *DTACK**. Alternatively, the slave asserts *BERR** if an error occurred.

5. The master responds by unasserting *DS**, which, in turn, directs the slave to unassert either *DTACK** or *BERR** as appropriate.

Note: ***Under no circumstances can a slave assert DTACK* and BERR* simultaneously.***

Block-Mode Data Transfers

A MXIbus master device can write to consecutive address locations on the MXIbus by means of a block-mode data transfer. A block-mode transfer is similar to a basic data transfer with some important differences, as described in the following sequence of steps:

1. The master device initiates the transfer by driving *AD[31–0]**, *PAR**, *WR**, *SIZE**, and *AM[4-0]** onto the MXIbus. *AD[31–0]** carries the desired address for the data transfer. The proper code on the *AM[4-0]** signals indicates the beginning of a block-mode transfer.

2. The master asserts *AS** to indicate that the address information on the MXIbus is valid. The slaves on the MXIbus latch this address into local counters. Each slave determines if the address is within its specified range. The slave being addressed will participate in the data transfer.

3. During a write operation, the master places data onto the MXIbus and asserts *DS**. The slave then latches the data. During a read operation, the master asserts *DS** to instruct the slave to place data onto the MXIbus. During both read and write operations, the slave responds to the master by toggling either *DTACK** to indicate a successful data transfer or *BERR** to indicate an error condition. The master continues to assert *AS** to indicate that additional transfers are forthcoming.

4. Once the slave asserts *DTACK**, the slaves on the MXIbus increment the address in their counters and step 3 is repeated.

5. The block-mode transfer terminates when the master unasserts *AS** during the last data transfer. The responding slave completes the current data transfer by asserting either *DTACK** or *BERR**.

Because all MXIbus slaves monitor the data transfer, it is possible to write to more than one slave device during a block-mode transfer; simply perform a block-mode transfer over an address range covered by more than one slave. Because the address counters on all the slaves increment, slaves respond in succession to data transfers as the address counters enter the valid address space of each slave. If a slave asserts *BERR**, the block-mode transfer terminates immediately.

Indivisible Data Transfers

An indivisible data transfer is virtually identical to a block-mode data transfer; however, the consecutive read or write operations are directed at the same address location. In other words, the slaves do not increment their address counters at the completion of each data transfer. To enable this mode, a master should place address modifier codes that are associated with basic data transfers onto the MXIbus, then assert *AS** over consecutive data transfers.

Priority Select Data Transfers

The final type of data transfer is the priority select transfer. Master and slave devices have the option of implementing this type of transfer, which all MXIbus System Controllers are required to accommodate. It is similar to the basic data transfer and is useful in situations where multiple slaves service the same address range. A priority select transfer proceeds as follows:

1. The master device initiates the transfer by driving *AD[31–0]**, *PAR**, *WR**, and *SIZE** onto the MXIbus. *AD[31–0]** carries the desired address for the data transfer. The master places a priority select address modifier code onto the *AM[4–0]** signals. One or more slave(s) recognize that the address is within its defined address region.

2. The master asserts *AS** to inform the slaves that the address information on the MXIbus is valid. The slaves then latch the address information.

3. To determine which slave will respond, the MXIbus System Controller initiates bus arbitration when the master asserts *DS**. If more than one slave not only maps to the address but also wants to participate in the data transfer, the MXIbus System Controller grants the data transfer to the slave with the highest priority. During a write operation, the master places data onto the MXIbus and asserts *DS**. The slave then latches the data. During a read operation, the master asserts *DS** to instruct the slave to place data onto the MXIbus. At some point during this part of the data transfer, the master unasserts *AS**.

4. To complete the data transfer, the master device unasserts *DS**. The slave performing the transfer responds by asserting *DTACK** if the transfer was successful or by asserting *BERR** if an error occurred.

5. The master responds by unasserting *DS**, which directs the slave to unassert either *DTACK** or *BERR** as appropriate.

Interrupts

The MXIbus has capabilities to handle interrupts. For this purpose, you need to designate a single device as an interrupt handler. When a device on the MXIbus asserts the *IRQ** signal, the interrupt-handling device performs a priority select data transfer to receive an interrupt vector. Because the content of the interrupt vector is not dictated by the MXIbus specification, you have considerable flexibility in configuring interrupt schemes. Notice that the *IRQ** signal gives you the means to generate interrupts, while the priority select data transfer protocol gives you the means to impose interrupt priority levels. How you can use these features depends on the intended application.

Relationship between the MXIbus, VXIbus, and MC-MXI

One of the driving motivations behind the MXIbus was the goal to link general-purpose computers to VXIbus devices and instrumentation. The MXIbus is primarily an electrical specification that defines signals and time relationships. The VXIbus, however, is both an electrical specification and an architectural specification. Elements of the VXIbus architecture are incorporated into many MXIbus interface products, including the MC-MXI, so that MXIbus

devices can better communicate with VXIbus devices. This section briefly describes those architectural elements of the VXIbus that MXIbus devices commonly implement. The most useful VXIbus architectural elements are as follows:

- Configuration registers
- VXI device class
- Word Serial Protocol communication

Not all MXIbus devices implement these functions. In fact, some parts of these architectural elements are mutually exclusive.

Configuration Registers

All VXIbus devices must include four configuration registers. The four registers are mapped into a portion of the upper 4 KB of A16 space. The four registers are as follows:

- The *Identification/Logical Address* register defines the logical address of the VXIbus device. A logical address is a unique 8-bit number from 0 to 255 that identifies the VXIbus device. The logical address determines the base address of the configuration registers in A16 space. The following formula determines the base address:

  Base Address = C000h + LA * 40h

  where *LA* is the logical address assigned to the VXIbus device. In addition to the logical address, this register contains information such as the manufacturer of the device and the device class. This register also determines if the VXIbus device will recognize either A24 space or A32 space. While there is a choice between A24 and A32 space, the requirements for the configuration registers force all VXIbus devices to exist in A16 space.
- The *Device Type* register contains a unique identification number for the VXIbus device.
- The *Control/Status* register contains critical information on the state of the VXIbus device.
- The *Offset* register defines the base address of the VXIbus device in either A24 or A32 space, depending on the contents of the Identification/Logical Address register.

VXIbus Device Classes

VXIbus devices fall into four different classes. The device's behavior depends on its device class. The most apparent difference among the device classes is the number and type of registers that each device class implements. The four device classes are as follows:

- *Register-Based* devices implement the configuration registers as described in the previous section and implement nothing else. They are the simplest VXIbus devices.
- *Memory-Based* devices can be RAM, ROM, or other memory devices. Most MXIbus interface products do not use this class.
- *Message-Based* devices have a register set for transmitting simple commands. This device class also accommodates shared memory data transfers in which local memory on a VXIbus device is accessible to other VXIbus devices in either A24 or A32 space. Most MXIbus devices belong to this class.

- *Extended* Register-Based or Memory-Based devices are either Register-Based devices or Memory-Based devices that have an extended set of configuration registers.

Word Serial Protocol

Word Serial Protocol gives VXIbus devices the capability to communicate by using simple messages. The objective is similar to that of Message-Based devices. In fact, many Message-Based devices support Word Serial Protocol to complement the required capabilities of Message-Based devices. There is a great deal of flexibility in implementing hardware support for Word Serial Protocol. The National Instruments VXIbus devices and MXIbus interface products have a 16-bit-wide by 8-word-deep FIFO (called the signal register) and data registers for Word Serial Protocol communications. Application software determines the types of commands to transmit. For some specified common commands, you can also refer to an extension of the VXIbus specification, VXI-3, Revision 1.0, *Word Serial Commands for Version and Serial Number Identification Specification*.

MXIbus Performance

It is often difficult to understand how a performance specification for a single component relates to the overall performance of your system. In the case of MXIbus, it is important to understand not only the performance issues associated with the MXIbus link but also the devices that communicate across the link.

Comparing Data Transfer Rates

A common benchmark for VXIbus and MXIbus devices is the *Block Data Rate*. This benchmark is easy for vendors to isolate and measure under ideal conditions. It is important to understand what the Block Data Rate means to your application. Block Data Rate is the rate at which you can move a large block of data to or from memory on an ideal MXIbus device by using back-to-back MXIbus transfers. Block Data Rate does not measure how fast the MXIbus device can process the blocks of data, or how fast the MXIbus device can store them to disk once they are moved, or whether your MXIbus devices themselves are actually capable of that data transfer rate. Most applications are not limited by the Block Data Rate of the VXIbus interface hardware: they are limited by the total time required to both move and handle the data or by the rate at which the instruments themselves can generate or accept the data.

Block Data Rate is easy for vendors to specify, but it is often difficult for users to relate to overall system performance. It is only one of many elements that affect the actual throughput of your system. For example, consider the system configuration shown in Figure 1-5. In this system, a Micro Channel–based computer (CPU 1) is connected to the MXIbus via an MC-MXI, which in turn is connected via a VME-MXI interface to the VMEbus. The VMEbus, in addition, contains an embedded computer (CPU 2).

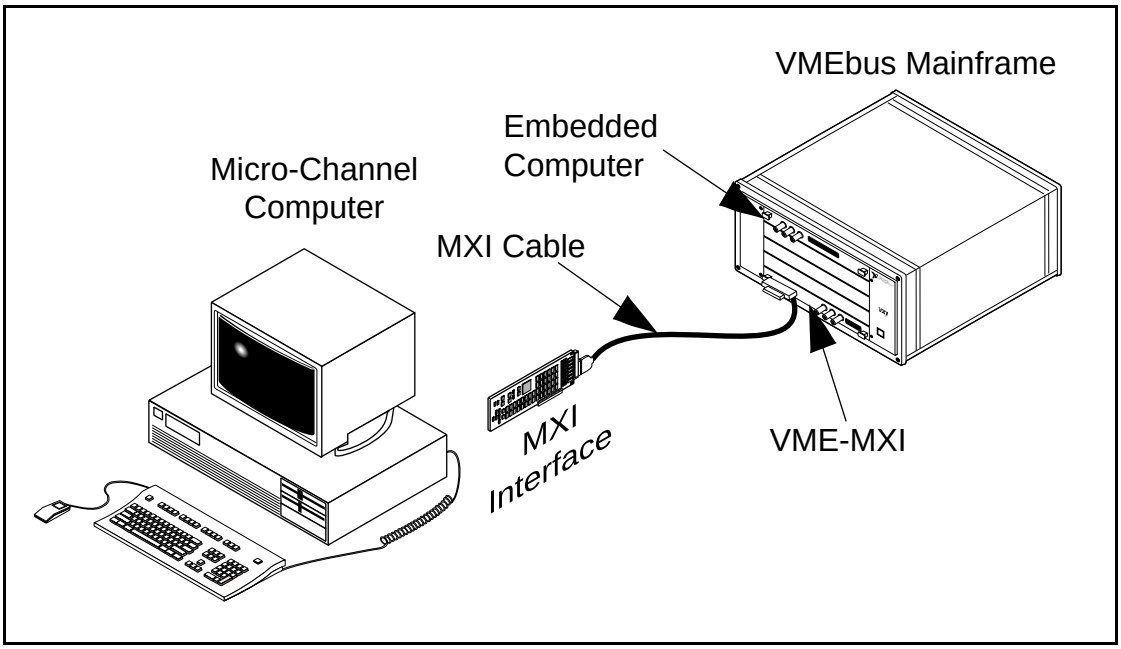


Figure 1-5. Typical MXIbus Application

When determining the data transfer rates from CPU 1 to CPU 2, you must consider many delay times, including the following:

1. Delay through the I/O circuitry of CPU 1 or, during block data transfers, the recovery time of CPU 1 from the previous data transfer
2. Propagation delay through the MC-MXI
3. Propagation delay across the MXIbus (10 ns/m)
4. Propagation delay through the VME-MXI
5. Access time to and from the I/O circuitry of CPU 2

The Block Data Rate accounts for delays attributable to items 2, 3, and 4. Notice that the Block Data Rate is only a part of overall system performance and, in most cases, is not the limiting component.

The theoretical maximum Block Data Rate for MXIbus is 20 MB/s. However, to obtain a more realistic estimate of system performance, you need to examine all five delay components listed in the example. All National Instruments MXIbus user manuals contain a *Specifications* appendix, which lists both the single (random access) and block transfer rates for its respective devices. You can think of these values as propagation delays, as listed in items 2 and 4 in the example, and use them to help calculate the transfer time of your system.

To estimate the MXIbus cycle time (and ultimately the data transfer rate), add the appropriate master rating of the device that will initiate the MXIbus transfer (item 2) to the appropriate slave rating of the device that will accept the MXIbus transfer (item 4). Also add the recovery time of the local CPU (item 1), the read/write access time of your remote system (item 5), and the length of your MXIbus cable (item 3). To determine the actual data transfer data rate, use the following equation:

$$\text{Data Transfer Rate (bytes/s)} = \frac{\text{Transfer Width (bytes)}}{\text{Transfer Time (s)}}$$

where *Transfer Width* equals the number of bytes per transfer, and *Transfer Time* equals the sum of all five items.

For example, consider the National Instruments VME-MC6000 kit. The MXIbus Master Mode time of the MC-MXI is 293 ns for block reads, and the MXIbus Slave Mode time of the VME-MXI is 238 ns for block reads. Therefore, if your actual application uses a 10-m MXIbus cable (90-ns MXIbus cable propagation time) and your VME device has a bus access time of 100 ns, the total transfer time for a single read during a block is 631 ns (assuming a 0-ns recovery time for the local system).

Assuming that your VME device is a 32-bit (4 bytes/transfer) device, your expected Block Read Data Rate to that VME device using the VME-MC6000 is 6.34 MB/s, as calculated by the following formula:

$$\text{Data Transfer Rate} = \frac{\text{4 bytes/transfer}}{\text{631 ns/transfer}} = \text{6.34 MB/s}$$

Local Performance

The MXIbus does not degrade the local performance of any MXIbus device when the MXIbus device is performing operations that do not involve the MXIbus. In those cases, when the MXIbus device performs a read or write that maps to a remote MXIbus device, the MXIbus hardware on both devices interlocks the bus cycle across the MXIbus to accomplish the transfer. This interlocking may affect local system performance.

$$\text{Data Transfer Rate} = = \text{6.34 MB/s}$$

Chapter 2
General Information

This chapter contains an overview of the functionality of the MC-MXI interface board, shows a picture of the MC-MXI board, and lists the contents of your kit and the available optional equipment.

The MC-MXI is an interface board that links a Micro Channel computer directly to the MXIbus. It uses address mapping to translate bus cycles on the Micro Channel bus to the MXIbus and vice versa. Figure 2-1 shows the MC-MXI interface board.

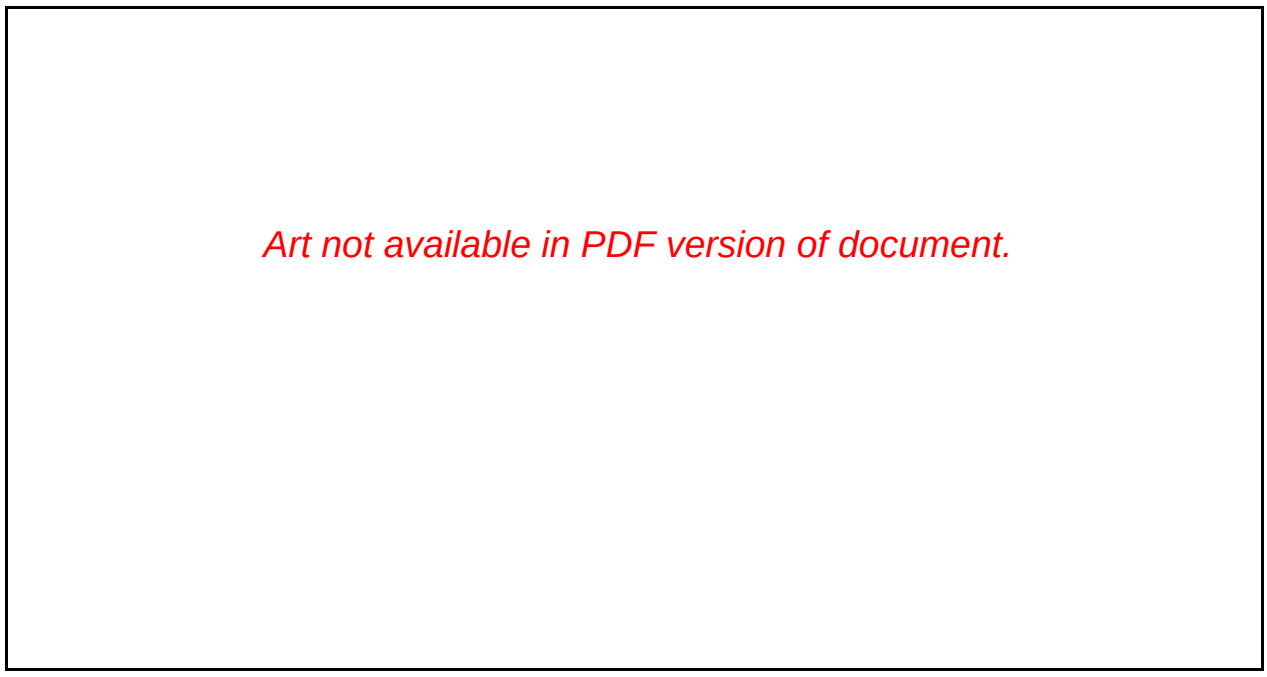


Figure 2-1. MC-MXI Interface Board

Overview

The MC-MXI can function as both a MXIbus master and a MXIbus slave. When operating as a MXIbus master, the MC-MXI converts Micro Channel memory cycles initiated by the CPU or an alternate bus master on the Micro Channel into MXIbus cycles intended for a MXIbus slave device. When operating as a MXIbus slave, the MC-MXI converts MXIbus cycles initiated by a MXIbus master into Micro Channel memory or I/O bus cycles so that other MXIbus devices can freely access (share) resources on the Micro Channel.

The MC-MXI implements many features of the MXIbus and the VXIbus, including the following:

- Data transfers in 8-bit, 16-bit, or 32-bit-wide data paths
- Capable of operating as a MXIbus master or MXIbus slave
- MXIbus System Controller
- Fully compatible VXI Message-Based device
- Extensive register set visible both in the Micro Channel I/O address space and in the MXIbus A16 address space
- Hardware support for Word Serial Protocol communication
- Up to three independent windows in the MXIbus A24 and A32 address spaces for MXIbus master operations
- Support for block mode, priority select, and indivisible data transfers
- Implementation of fairness and bus lock operations

The MC-MXI accesses MXI address spaces through one window when it is installed in a computer running DOS. In computers that do not run DOS, the MC-MXI uses three windows to access MXI address space. The MC-MXI's capabilities as a MXI device encompass part of its features. As a Micro Channel interface, the MC-MXI supports the following Micro Channel features:

- Two independent DMA channels: one to control data transfers from the Micro Channel to the MXIbus and the other to control data transfers from the MXIbus to the Micro Channel
- The ability for each DMA channel to be set with independent arbitration levels and burst-mode capabilities
- Four different types of interrupts, which can be mapped to Micro Channel interrupt levels 5, 10, 11, and 15
- Full implementation POS registers 0 to 5
- Eight-bit, 16-bit, and 32-bit data transfers
- Micro Channel master-mode and slave-mode operating capability

What Your Kit Should Contain

Your MC-MXI kit should contain the following components:

Kit Component	Part Number
MC-MXI Interface Board	181075-01
MC-MXI User Manual	320297-01

Optional Equipment

Equipment	Part Number
Type M1 MXIbus Cables Straight Point-to-Point Connectors:	
– 1 m	180758-01
– 2 m	180758-02
– 4 m	180758-04
– 8 m	180758-08
– 20 m	180758-20
Type M2 MXIbus Cables Straight Point-to-Right Angle Daisy-Chain Connectors:	
– 1 m	180760-01
– 2 m	180760-02
– 4 m	180760-04
– 8 m	180760-08
– 20 m	180760-20
Type M3 MXIbus Cables Right Angle Point-to-Right Angle Daisy-Chain Connectors:	
– 1 m	180761-01
– 2 m	180761-02
– 4 m	180761-04
– 8 m	180761-08
– 20 m	180761-20
MXIbus Terminating Pac (External)	180780-01

Optional Software

Your MC-MXI is shipped without interface software. This manual contains complete instructions for programming the MC-MXI directly. You can order various software packages from National Instruments to program and control the MC-MXI. The NI-VXI bus interface software package is a standardized set of utilities and C library functions that provides simple, low-level access to other MXIbus devices. NI-VXI is available across many different operating system platforms.

Software	Part Number
NI-VXI DOS Software for the MC-MXI	776419-11
NI-VXI Windows Software for the MC-MXI	776531-01
NI-VXI UNIX Software for the MC-MXI and AIX	776460-01
NI-VXI OS/2 Software for the MC-MXI	776421-01

You can use your MC-MXI with LabVIEW, which is an innovative, graphical programming system that is ideally suited for VXI applications. With LabVIEW, you can create and use software modules called virtual instruments (VIs), instead of writing programs. LabVIEW has all the power of a standard programming language, but is much easier to use because of its graphical environment.

If you want to use LabVIEW with your MC-MXI, you must order the NI-VXI Windows Software for the MC-MXI and the LabVIEW for Windows VXI Development System. It contains the subcomponents shown in the following table:

Software	Part Number
LabVIEW for Windows VXI Development System: LabVIEW for Windows Full Development System, LabVIEW for Windows VXI Library, LabVIEW for Windows/Sun VXI Instrument Library	776674-01

LabWindows is a development system in DOS for C and BASIC programmers. It contains extensive libraries of functions for data acquisition, data analysis, and data presentation, along with tools for editing and debugging your programs.

If you want to use LabWindows with your MC-MXI, you must order the NI-VXI DOS Software for the MC-MXI and the LabWindows for DOS, VXI Development System. It contains the subcomponents shown in the following table:

Software	Part Number
LabWindows for DOS, VXI Development System: LabWindows for DOS Full Development System, LabWindows for DOS VXI Libraries, LabWindows for DOS VXI Instrument Library	776729-01

Chapter 3
Configuration and Installation

This chapter describes the procedures for unpacking, configuring, and installing your MC-MXI interface board.

Unpacking

Follow these steps when unpacking your MC-MXI:

1. Before attempting to configure or install the MC-MXI, inspect the shipping container and its contents for damage. If damage appears to have been caused in shipment, file a claim with the carrier. Retain the packing material for possible inspection and/or for reshipment.

2. Verify that the pieces contained in the package you received match the kit parts list. *Do not* remove the board from its plastic bag at this point.

3. Your MC-MXI board is shipped packaged in an antistatic plastic bag to prevent electrostatic damage to the board. Some of the circuitry on the MC-MXI uses CMOS technology and can be damaged by electrostatic discharge. Before removing the board from the antistatic bag, touch the bag to a metal part of your computer chassis.

4. As you remove the MC-MXI from its bag, be sure to handle the board only by its edges. Avoid touching any of the IC components or connectors. Inspect the board for loose components or any other sign of damage. Notify National Instruments if the board appears damaged in any way. *Do not* install equipment that appears to be damaged.

Configuration

The only configurable hardware option on the MC-MXI is the ability to terminate the MXIbus signals on the interface board. Alternatively, you can use an external add-on module for MXIbus signal termination to aid in easy system reconfiguration. Only the first and last devices in the MXIbus daisy-chain should have terminating resistor networks.

The onboard termination option lets you install or remove the terminating resistor networks from sockets on the MC-MXI board. The board is shipped from the factory with these terminating resistor networks installed. If your MC-MXI is to be the first or last device in the MXIbus daisy-chain and you will *not* be using external terminating resistor networks, leave these internal resistor terminators in place. If the MC-MXI is not going to be an end device on the MXIbus daisy-chain or if you *will* be using external terminating resistor networks, remove the internal terminating resistor networks from their sockets. Store them in a safe place in case the MXIbus system configuration changes. When reinstalling the resistor networks, be sure to note the

position of pin 1 of the socket and the terminators, and make sure that the terminating networks are plugged firmly into their respective sockets.

Figure 3-1 shows the position of the six MXIbus terminating networks. All six networks must be either installed or removed from their sockets.

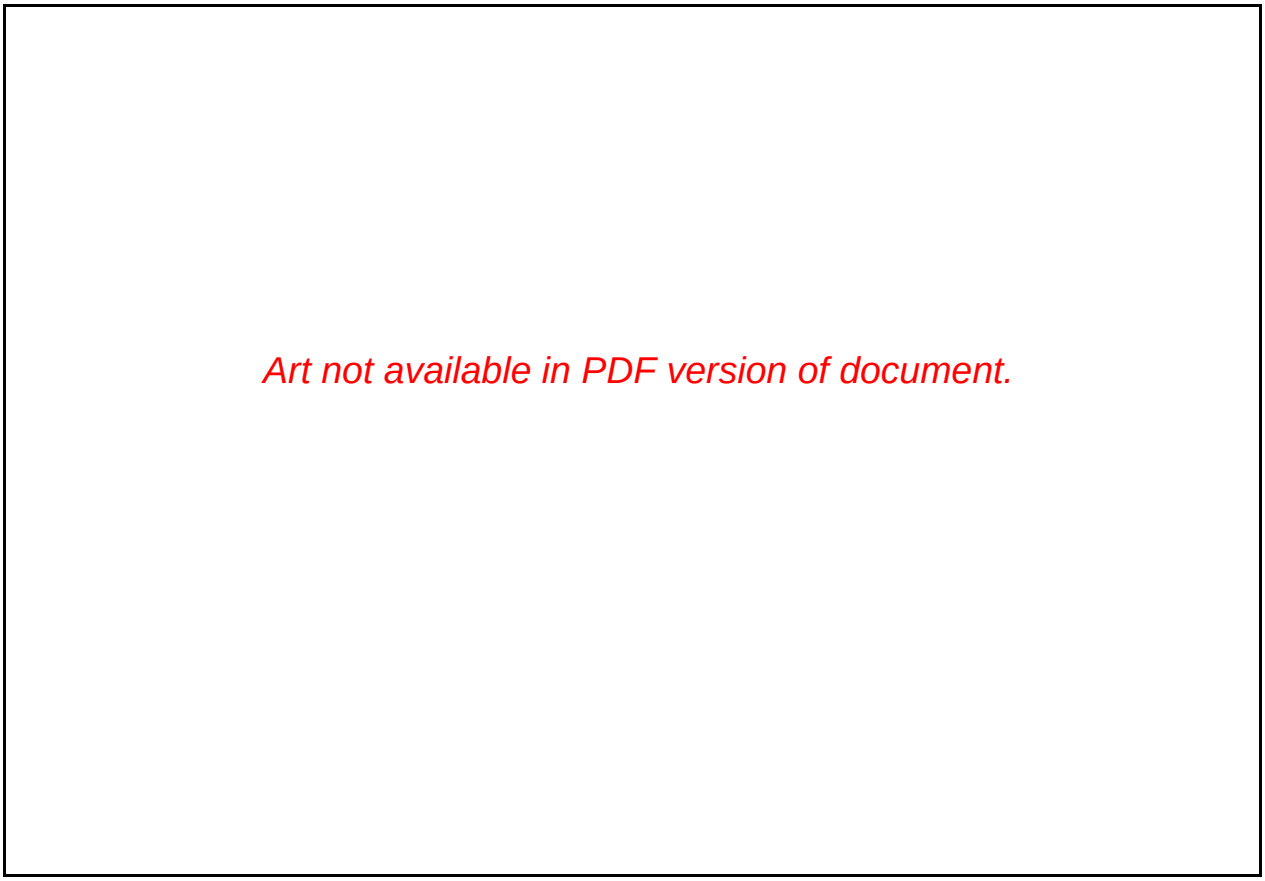


Figure 3-1. MXIbus Terminating Networks

Installation

Before attempting to install the MC-MXI, notice that the standard MXIbus cable connector is slightly wider than most cable connectors and might interfere with other cable connectors installed in adjacent slots. Normally, this will be a problem only if the cable connectors for the adjacent slots are also oversized. When choosing a Micro Channel slot to install the MC-MXI, verify that the MXIbus cable connector will not interfere with cables and connectors in other Micro Channel slots. If necessary, reposition other boards in the system to prevent cabling conflicts. It may also help to install the MC-MXI in one of the end slots so that you will have to contend with the cable connectors of only one other board.

If you cannot configure the MC-MXI to co-exist in an existing Micro Channel system by repositioning the cards, you can use one of the MXIbus cable options with a straight-point connector hood on the cable end that attaches to the MC-MXI. Because the straight-point connector hood is narrower than the MXIbus dual-connector arrangement, it gives an easier fit for many system configurations. However, this approach requires that the MC-MXI be the first device in the MXIbus daisy-chain, because a cable with a straight-point connector end cannot accept another MXIbus cable to propagate the bus. Remember that the first device in the MXIbus daisy-chain must also be configured as the MXIbus System Controller.

The following instructions are general installation instructions. Consult the user or technical reference manual of your computer for specific instructions and warnings.

1. Disconnect power to the computer.

2. Remove the top cover or access port to the Micro Channel I/O bus.

3. Select a 32-bit, full-length expansion slot or a full-length, 32-bit matched memory slot. These slots have a longer card edge expansion connector than those found on 16-bit slots. Most Micro Channel computers have an assortment of both 16-bit and 32-bit expansion slots. A 32-bit board, such as the MC-MXI, cannot be installed in a 16-bit expansion slot.

4. Locate, for the slot you have selected, the metal bracket that covers the cut-out in the back panel of the Micro Channel chassis. Loosen the bracket-retaining thumbscrew and remove the bracket by lifting it out of the top of the slot.

5. Line up the MC-MXI with the MXIbus connector that is near the cut-out on the back panel and the plastic card guide that is lined up with the respective slot guide. Slowly push down on the front of the MC-MXI until its card edge connector is resting on the expansion slot receptacle. Using slow, evenly distributed pressure, press the MC-MXI straight down until it sits in the expansion slot.

6. Tighten the bracket-retaining thumbscrew to secure the MC-MXI to the back panel rail.

7. Check the installation, then replace the computer cover.

8. Connect the MC-MXI to the rest of your MXIbus system by connecting the proper MXIbus cable(s) to the back of the board.

9. Reconnect power to the computer.

Chapter 4
Register Descriptions

This chapter contains information on the use of the MC-MXI registers.

Overview

The MC-MXI registers control all user-configurable functions on the MC-MXI. This is in contrast to some interface boards that use a combination of registers, jumpers, and switches to control operations. The MC-MXI incorporates three register sets. These register sets and the functions they perform are defined below:

- MXIbus registers implement the required registers for MXI devices and VXI Message-Based devices.
- MC-MXI local registers store control and status information specific only to the MC-MXI.
- POS registers implement the configuration register set required by the Micro Channel as specified by IBM.

The MC-MXI register set is implemented with three large-scale components:

- The SMC 94C18 Micro Channel controller chip implements the mandatory logic functions of the Micro Channel bus and contains support for the POS registers and some local configuration registers.
- The National Instruments Message-Based Interface Gate Array (MIGA) implements the MXIbus registers.
- The Intel 82C54 Peripheral Interface Timer (PIT) contains registers that establish time limits for Micro Channel and MXIbus operations.

Cross-Referencing Registers

The descriptions of the individual registers reference other registers in the MC-MXI. Most register descriptions include a *See Also* section that lists the registers referenced in the body of the register description.

Definitions

When this manual describes the individual bit fields in the registers, the term *set* means setting the value of the bit to 1. The term *clear* means setting the value of a bit to 0. When describing

the register type, *r* means that the register is read only, *w* means that the register is write only, and *r/w* means that the register has both capabilities. Appendix B, *Mnemonics Key*, defines the mnemonics that this manual uses to describe register and bit names; refer to the glossary for other definitions. An asterisk (*) after the name of a signal or bit field indicates that the signal or bit field is active low. An asterisk is equivalent to an overbar.

Register Bit Fields with Constant Values

Some bit fields have a constant value of 0 or 1. When writing to these bits, always write the specified constant value. When reading these bit fields, the MC-MXI will return the specified constant value.

Register Bit Fields with Undefined Value

Some bit fields have an undefined value. These bit fields are denoted with an *X*. Writing to these bits does not affect the operation of the MC-MXI and reading these bits returns either a 0 or a 1.

Read-Only and Write-Only Register Bit Fields

In some read/write registers, some bit fields are read only or write only. In these situations, writing to a read-only bit does not affect the operation of the MC-MXI and reading from a write-only bit returns an undefined value.

Scope of Resets

The MC-MXI employs a hierarchical reset structure, which is described in Chapter 6, *Theory of Operation*. Different reset levels affect different individual bit fields in the registers.

Register Byte Ordering

When accessing 16-bit registers from the Micro Channel, the least significant byte is located at an *even* address and the most significant byte is located at an *odd* address. However, when these registers are accessed from the MXIbus, the most significant byte is located at an even address and the least significant byte is located at an odd address.

Register Map

Table 4-1 shows the register map for the MC-MXI registers. The table gives the register name, the address, the size in bits, and the type of register. The table shows the register names for both the read and write modes, as appropriate. An *X* in the *MXI Acc.* column indicates that a register is accessible from the MXIbus, and an *X* in the *MC Acc.* column indicates that a register is

accessible from the Micro Channel. These registers have different base addresses, depending on whether they are accessed from the MXIbus or from the Micro Channel, but the offset from the base address for each register is the same for both the MXIbus and the Micro Channel.

Note: ***Some registers have different addresses for read and write operations.***

The local CPU assigns the Micro Channel I/O base address for the MC-MXI registers during power-up configuration. The local CPU reads the Adapter Descriptor File (ADF) that came with your MC-MXI and configures the board I/O and memory base addresses by writing to POS register 103h.

Most of the registers are also available to the MXIbus in the MXI A16 address space. The base address of the MC-MXI registers with respect to the MXIbus can be determined by the formula

I/O base address = C000h + (LA * 40h)

where *LA* is the logical address of the MC-MXI.

As noted in the previous section, the registers have different byte orderings, depending on whether they are accessed from the Micro Channel or from the MXIbus. Because the MC-MXI implements VXI Message-Based device registers, it must conform to the same byte ordering as other VXI devices when it is accessed from the MXIbus. Because the Micro Channel uses a different byte ordering from the VXIbus, the high and low bytes for the registers are not ordered in the same way. The 0 and 1 designations for the individual bytes in the register descriptions represent the Micro Channel byte-ordering convention. Refer to the *Byte Swapping* section in Chapter 5, *Programming the MC-MXI*, for more information on byte ordering.

The MC-MXI register set has several different registers that are referred to as *status* or *control* registers. These status and control registers operate separately on the MXIbus, Micro Channel, or the internal circuitry of the MC-MXI. Carefully examine the register descriptions to understand the scope of the different control and status registers.

Table 4-1. MC-MXI Register Map

Register Name Read Mode	Register Name Write Mode	MXI Acc.	MC Acc.	Offset from Base Address (Hex)	Type	Size
Basic MXIbus Registers:						
MXI ID	MXI Logical Addr.	X	X	0	R/W	16/8
Device Type		X	X	2	R	16/8
MXI Status	MXI Control	X	X	4	R/W	16/8
MXI Offset	MXI Offset	X	X	6	R/W	16/8
Message-Based Device Registers:						
MXI Protocol	MXI Signal	X	X	8	R/W	16/8
MXI Response	Data Extended	X	X	A	R/W	16/8
Data High (Out)	Data High (In)	X	X	C	R/W	16/8
Data Low (Out)	Data Low (In)	X	X	E	R/W	16/8
A24 Pointer (Low)	A24 Pointer (High)	X	X	10	R/W	16/8
A24 Pointer (High)	A24 Pointer (Low)	X	X	12	R/W	16/8
A32 Pointer (Low)	A32 Pointer (High)	X	X	14	R/W	16/8
A32 Pointer (High)	A32 Pointer (Low)	X	X	16	R/W	16/8
MXI Logical Addr.	MXI ID	X	X	20	R/W	16/8
	Device Type	X	X	22	W	16/8
Soft Reset Service	MXI Status	X	X	24	W	16/8
Comm. Status	Comm. Control	X	X	26	R/W	16/8
MXI Signal	MXI Protocol	X	X	28	R/W	16/8
Data Extended	MXI Response	X	X	2A	R/W	16/8
Data High (In)	Data High (Out)	X	X	2C	R/W	16/8
Data Low (In)	Data Low (Out)	X	X	2E	R/W	16/8
Slave Configuration	Slave Configuration	X	X	30	R/W	16/8
MC-MXI Local Registers:						
PITLOCAL	PITLOCAL		X	48	R/W	8
PITREADY	PITREADY		X	4A	R/W	8
PITSC	PITSC		X	4C	R/W	8
	PITCNTR		X	4E	W	8
	WIN1AM-SMOFF		X	50	W	16/8
	WIN3AM-WIN2AM		X	58	W	16/8
Master Mode PAGE	Master Mode PAGE		X	60	R/W	16/8
MC-MXI–STATUS	MC-MXI–CTRL		X	68	R/W	16/8
WAITSTATES	WAITSTATES		X	140	R/W	8
DMA Enable	DMA Enable		X	141	R/W	8
Interrupt Enable	Interrupt Enable		X	142	R/W	8
MC INT Status	MC INT Ack.		X	143	R/W	8
SCCLK	SCCLK		X	144	R/W	8
DMA Control	DMA Control		X	145	R/W	8

The Micro Channel specification requires adapter cards to support a set of Programmable Option Select (POS) registers in a special address space (0–7 hex). Table 4-2 lists the registers located in POS address space. Notice that the MC-MXI does not support registers at addresses 6 and 7.

Table 4-2. POS Register Map

Register Name	Address (Hex)	Read/Write
MC-MXI Micro Channel ID (Low)	0	Read
MC-MXI Micro Channel ID (High)	1	Read
MC Setup/DMA Channel 1	2	Read/Write
Memory/IO Base Address	3	Read/Write
MC Interrupt Map	4	Read/Write
MC Arbitration/DMA Channel 2	5	Read/Write
Not implemented	6	
Not implemented	7	

Note: ***These registers are accessible only from the Micro Channel.***

Basic MXIbus Registers

MXI ID Register

Function: This register stores information about the MC-MXI device class, the address space it uses, and the identification number of the manufacturer.

Address:

Micro Channel Read:	I/O base address + 0h
Micro Channel Write:	I/O base address + 20h
MXIbus A16 Space Read:	C000h + (LA * 40h) + 0h
MXIbus A16 Space Write:	C000h + (LA * 40h) + 20h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
CLASS[1]	CLASS[0]	ADSP[1]	ADSP[0]	MANID[11]	MANID[10]	MANID[9]	MANID[8]	1
7	6	5	4	3	2	1	0	
MANID[7]	MANID[6]	MANID[5]	MANID[4]	MANID[3]	MANID[2]	MANID[1]	MANID[0]	0

Bit	Mnemonic	Description
15–14r/w	CLASS[1–0]	Device Class Bit Field

These bits indicate the classification of the MC-MXI according to the following table. These bits are not affected by a hard or soft reset. The MC-MXI is a Message-Based device; you must write 10b to these bits.

CLASS[1–0]	Device Class
00b	Memory
01b	Extended
10b	Message-Based
11b	Register-Based

Bit	Mnemonic	Description
13–12r/w	ADSP[1–0]	Address Space Bit Field

These bits indicate the addressing mode(s) of the MC-MXI communication registers and memory according to the following table. These bits are not affected by a hard or soft reset.

ADSP[1–0]	Address Space
00b	A16/A24
01b	A16/A32
10b	Reserved
11b	A16 Only

11–0r/w	MANID[11–0]	Manufacturer ID Bit Field

This field uniquely identifies the manufacturer of the MC-MXI. The identification for National Instruments, FF6h, should be written here during initialization. These bits are not affected by a hard or soft reset.

MXI Logical Address Register

Function: The content of the Logical Address register determines the location of the MC-MXI registers within MXI A16 space.

Address:

Micro Channel Read:	I/O base address + 20h
Micro Channel Write:	I/O base address + 0h
MXIbus A16 Space Read:	C000h + (LA * 40h) + 20h
MXIbus A16 Space Write:	C000h + (LA * 40h) + 0h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
0	0	0	0	0	0	0	0	1
7	6	5	4	3	2	1	0	
LA[7]	LA[6]	LA[5]	LA[4]	LA[3]	LA[2]	LA[1]	LA[0]	0

Bit	Mnemonic	Description
15–8	0	Reserved Bits These bits are reserved for future use and return a meaningless value. Always write 0 to these bits.
7–0r/w	LA[7–0]	Logical Address Bit Field The logical address field contains the logical address of the MC-MXI. The MC-MXI registers appear in MXIbus A16 space at C000h + (40h * LA[7–0]). These bits are not affected by a hard or soft reset.

MXI Device Type Register

Function: The Device Type register controls the amount of A24 or A32 space required by the MC-MXI and stores the model code for the MC-MXI.

Address:

Micro Channel Read:	I/O base address + 2h
Micro Channel Write:	I/O base address + 22h
MXIbus A16 Space Read:	C000h + (LA * 40h) + 2h
MXIbus A16 Space Write:	C000h + (LA * 40h) + 22h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
RQMEM[3]	RQMEM[2]	RQMEM[1]	RQMEM[0]	MDL[11]	MDL[10]	MDL[9]	MDL[8]	1
7	6	5	4	3	2	1	0	
MDL[7]	MDL[6]	MDL[5]	MDL[4]	MDL[3]	MDL[2]	MDL[1]	MDL[0]	0

Bit	Mnemonic	Description
15–12r/w	RQMEM[3–0]	Required Memory Bit Field This field specifies the amount of MXIbus A24 or A32 memory space to be mapped into Micro Channel memory space. The following equation illustrates how to use this field. In the equation, the letter *a* represents the value of the Address Space bit field (ADSP[1–0]) in the MXI ID register, and the letter *m* is the value of RQMEM. These bits are not affected by a hard or soft reset. $256^{a} * 2^{23-m}$
11–0r/w	MDL[11–0]	Model Code Bit Field This field uniquely identifies the MC-MXI. The model code for the MC-MXI is FF5h and should be written here during initialization. These bits are not affected by a hard or soft reset.

See Also: MXI ID Register, MXI Control Register

MXI Status Register

Function: The MXI Status register indicates the status of the MC-MXI to other devices on the MXIbus. Some of the bits in this register are read only.

Address:

Micro Channel Read:	I/O base address + 4h
Micro Channel Write:	I/O base address + 24h
MXIbus A16 Space Read:	C000h + (LA * 40h) + 4h
MXIbus A16 Space Write:	C000h + (LA * 40h) + 24h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
MEMACT	X	RESETSRC	STAT[12]	STAT[11]	STAT[10]	STAT[9]	STAT[8]	1
7	6	5	4	3	2	1	0	
STAT[7]	STAT[6]	STAT[5]	STAT[4]	READY	PASSED	X	RESET	0

Bit	Mnemonic	Description
15r	MEMACT	Memory Active Bit This read-only bit reflects the status of the MEMACTW bit in the MXI Control register. Writing to this bit has no effect.
14	X	Reserved Bit This read-only bit is reserved for future use and returns a meaningless value.
13r	RESETSRC	Reset Source Bit This read-only bit indicates the source of the last reset. If RESETSRC is 1, the last reset was a hard reset. If RESETSRC is 0, the last reset was a soft reset.
12–4r/w	STAT[12–4]	MXI Status Register Bits [12–4] These are general-purpose read/write bits. These bits are not used by the MC-MXI and can be written with any value and read back later. These bits are not affected by a hard or soft reset.

Bit	Mnemonic	Description
3r/w	READY	Device Ready Bit The local CPU should manipulate READY to reflect the state of the MC-MXI. Whether a 1 or 0 denotes a ready state is application specific. This bit is cleared on a hard or soft reset.
2r/w	PASSED	Device Passed Bit The local CPU should set this bit to 1 if the local CPU and the MC-MXI are operating properly. This bit is cleared on a hard or soft reset.
1	X	Reserved Bit This bit is reserved for future use and returns a meaningless value.
0r	RESET	Soft Reset Bit This read-only bit reflects the state of the RESET bit in the MXI Control register.

MXI Control Register

Function: The MXI Control register provides control bits for the Memory Enable and the MC-MXI soft reset.

Address: Micro Channel Write: I/O base address + 4h
MXIbus A16 Space Write: C000h + (LA * 40h) + 4h

Type: Write Only

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
MEMACTW	0	0	0	0	0	0	0	1

7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	RESET	0

Bit	Mnemonic	Description
15w	MEMACTW	Memory Active Bit Setting this bit enables the A24/A32 address decode function of the MC-MXI. This bit is cleared after a hard reset and is not affected by a soft reset.
14–1	0	Reserved Bits These bits are reserved for future use. Always write a 0 to these bits.
0w	RESET	Soft Reset Bit Setting this bit forces the MC-MXI into the soft reset state if the SRSTEN bit in the Communication Control register is set. This soft reset affects registers between 0 and 31 from the I/O base address of the MC-MXI. It does not affect other MC-MXI registers or circuitry. This bit is cleared after a hard reset.

See Also: Communication Control Register

MXI Offset Register

Function: This register defines the MXIbus base address in either A24 or A32 space that the MC-MXI maps into Micro Channel memory space.

Address:

Micro Channel Read:	I/O base address + 6h
Micro Channel Write:	I/O base address + 6h
MXIbus A16 Space Read:	C000h + (LA * 40h) + 6h
MXIbus A16 Space Write:	C000h + (LA * 40h) + 6h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
OFF[15]	OFF[14]	OFF[13]	OFF[12]	OFF[11]	OFF[10]	OFF[9]	OFF[8]	1
7	6	5	4	3	2	1	0	
OFF[7]	OFF[6]	OFF[5]	OFF[4]	OFF[3]	OFF[2]	OFF[1]	OFF[0]	0

Bit	Mnemonic	Description
15–0r/w	OFF[15–0]	MXIbus A24 or A32 Offset Bits The *m*+1 most significant bits of this register are the values of the *m*+1 most significant bits of the MXIbus base address that are mapped to Micro Channel space, where *m* is the value of the Required Memory bit field (RQMEM) in the MXI Device Type register. The 15-*m* least significant bits are meaningless. Thus, the Offset register bits 15 through 15-*m* must match the MXIbus address lines A_{23} through A_{23-m} for A24 space, or lines A_{31} through A_{31-m} for A32 space. See the *MXIbus Slave-Mode Operation* section in Chapter 5, *Programming the MC-MXI*, for more information and programming examples.

See Also: MXI Device Type Register

Message-Based Device Registers

MXI Protocol Register

Function: The Protocol register indicates various features that are implemented by the MC-MXI and the MXI application software.

Address:

Micro Channel Read:	I/O base address + 8h
Micro Channel Write:	I/O base address + 28h
MXIbus A16 Space Read:	C000h + (LA * 40h) + 8h
MXIbus A16 Space Write:	C000h + (LA * 40h) + 28h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
CMDR*	SIGREG*	MASTER*	INTRPTR	FHS*	SHMEM*	PROT[9]	PROT[8]	1
7	6	5	4	3	2	1	0	
PROT[7]	PROT[6]	PROT[5]	PROT[4]	PROT[3]	PROT[2]	PROT[1]	PROT[0]	0

Bit	Mnemonic	Description
15r/w	CMDR*	MXIbus Commander Bit Setting this bit indicates that the MC-MXI has Servant-only capability. Clearing this bit indicates that the MC-MXI has both Servant and Commander capability. This bit is not affected by a hard or soft reset.
14r/w	SIGREG*	MXIbus Signal Register Bit Clearing this bit indicates the MC-MXI has a functional Signal register. Setting this bit indicates that the MC-MXI does not have a Signal register. While the MC-MXI physically has a Signal register, the local CPU, by clearing or setting SIGREG*, can determine if the register will be recognized. This bit is not affected by a hard or soft reset.

Bit	Mnemonic	Description
13r/w	MASTER*	MXIbus Master Bit Clearing this bit indicates the MC-MXI has MXIbus master capability. Setting this bit indicates the MC-MXI does not have master capability. This bit is not affected by a hard or soft reset.
12r/w	INTRPTR	MXIbus Interrupter Bit Setting this bit indicates that the MC-MXI has interrupter capability. Clearing this bit indicates the MC-MXI will not, and cannot, send interrupts to the MXIbus. This bit is not affected by a hard reset or soft reset.
11r	FHS*	MXIbus Fast Handshake Bit A 1 in this bit indicates the MC-MXI supports normal Word Serial transfers as a servant. The MC-MXI does not support Fast Handshake. The FHS* bit is hard-wired internally to 1. Writing to this bit has no effect on the MC-MXI.
10r/w	SHMEM*	MXIbus Shared Memory Bit Clearing this bit indicates that the MC-MXI supports the shared memory protocol and implements one or both of the A24 Pointer and A32 Pointer registers. Setting this bit indicates that the MC-MXI does not support the shared memory protocol. This bit is not affected by a hard or soft reset.
9–4r/w	PROT[9–4]	Protocol Register Bits [9–4] The local CPU should initialize these bits to all 1s. These bits are not affected by a hard or soft reset.
3–0r/w	PROT[3–0]	Protocol Register Bits [3–0] These are general-purpose read/write bits and can contain any value. These bits are not affected by a hard or soft reset.

See Also: MXI Signal Register

MXI Signal Register

Function: The Signal register is used for general device-to-device signaling. The Signal register is implemented as a 16-bit-wide by 8-word-deep FIFO.

Address:

Micro Channel Read:	I/O base address + 28h
Micro Channel Write:	I/O base address + 8h
MXIbus A16 Space Read:	C000h + (LA * 40h) + 28h
MXIbus A16 Space Write:	C000h + (LA * 40h) + 8h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
SIG[15]	SIG[14]	SIG[13]	SIG[12]	SIG[11]	SIG[10]	SIG[9]	SIG[8]	1
7	6	5	4	3	2	1	0	
SIG[7]	SIG[6]	SIG[5]	SIG[4]	SIG[3]	SIG[2]	SIG[1]	SIG[0]	0

Bit	Mnemonic	Description
15–0r/w	SIG[15–0]	MXIbus Signal FIFO

These bits form the data word of the MXI Signal register. A write to this register sets the SIGREQ bit in the Communication Status register. Also, writing to the Signal register generates a Micro Channel interrupt if the COMMEN bit in the Interrupt Enable register and the SIGIE bit in the Communication Control register are set. The SIGREQ bit and the interrupt remain asserted until the MXI Signal register FIFO is empty. The SIGREQ bit is updated on writes to the lower 8 bits (SIG[7–0]) of the Signal register. Therefore, a MXI device must write two consecutive bytes by either locking the MXIbus, using two single-byte indivisible data transfers, or using a single 16-bit data transfer. Writing the two bytes consecutively prevents the MXI Signal register contents and flags from becoming corrupted.

As an example of what could go wrong, the following is a scenario in which two devices, A and B, each try to access the MXI Signal register of the MC-MXI with 8-bit writes. Problems can arise when Servants A and B perform the following operations:

1. Servant A acquires the MXIbus and writes the upper 8 bits of its signal to the MC-MXI.

2. Servant B acquires the MXIbus and writes the upper 8 bits of its signal to the MC-MXI.

3. Servant A acquires the MXIbus and writes the lower 8 bits of its signal to the MC-MXI.

4. Local CPU detects that the SIGREQ bit is set, reads the Signal register, and gets the upper 8 bits from Servant B and the lower 8 bits from Servant A.

If the MXI Signal register is full and a device attempts to write another signal into the FIFO (upper or lower byte, or both), the MC-MXI responds with a *BERR** and does not store the signal.

The MXI Signal register FIFO can store up to eight signal transactions. The master MXIbus device may, at its option, retry writing the signal at a later time. One signal transaction (that is, 16 bits) is removed from the FIFO each time the local CPU reads the MXI Signal register.

The MXI Signal register FIFO is initialized to an empty state on a hard or soft reset.

See Also: Communication Control Register, Communication Status Register, Interrupt Enable Register.

MXI Response Register

Function: This register provides the status of MXIbus communications. Some of the bits in this register are read only.

Address:

Micro Channel Read:	I/O base address + Ah
Micro Channel Write:	I/O base address + 2Ah
MXIbus A16 Space Read:	C000h + (LA * 40h) + Ah
MXIbus A16 Space Write:	C000h + (LA * 40h) + 2Ah

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
0	RESP[14]	DOR	DIR	ERR*	RDRDY	WRRDY	FHSACT*	1

7	6	5	4	3	2	1	0	
LOCKED*	DL[1]	DL[0]	MQEDET	RDERR	WRERR	RESP[1]	RESP[0]	0

Bit	Mnemonic	Description
15	0	Reserved Bit This read-only bit is hard wired to 0. Writing to this bit has no effect.
14r/w	RESP[14]	Response Register Bit [14] This bit is reserved and should always be written as 1. This bit is set on a hard reset and is not affected by a soft reset.
13r/w	DOR	Data Out Ready Bit This bit indicates the state of the output data path. A 1 indicates that the MC-MXI is ready to output data; a 0 indicates that it is not ready. This bit is cleared on a hard or soft reset.
12r/w	DIR	Data In Ready Bit This bit indicates the state of the input data path. A 1 in this field indicates that the MC-MXI is ready to accept data; a 0 indicates that it is not ready. This bit is cleared on a hard or soft reset.

Bit	Mnemonic	Description
11r/w	ERR*	Error Bit During Word Serial Protocol data transfers, the local CPU manipulates this bit to reflect error conditions. A 0 indicates that a Word Serial Protocol error, which was not reported via the *Read Protocol Error* query, occurred. A 1 indicates that there are no Word Serial Protocol errors. This bit is set on a hard or soft reset.
10r/w	RDRDY	Read Ready Bit A 1 indicates that the Data Out register(s) contain(s) data for a remote commander to read. RDRDY is automatically set to 1 when the local CPU writes to the low byte of the Data Low (Out) register. RDRDY is cleared during a read of the Data Low (Out) register. Normally, the Data Low (Out) register is read by a remote commander. This bit is cleared on a hard reset or soft reset.
9r/w	WRRDY	Write Ready Bit The local CPU sets this bit during Word Serial transfers after the local CPU reads a value from the Data Low (In) register. Writing data to the low byte of the Data Low (In) register automatically clears WRRDY. This bit is cleared on a hard reset or soft reset.
8r	FHSACT*	Fast Handshake Active Bit A 1 indicates that the MC-MXI supports only normal Word Serial transfers as a servant. The MC-MXI does not support Fast Handshake and the FHSACT* bit is hard wired internally to 1. Writing to this bit has no effect.
7r/w	LOCKED*	MXIbus Servant Lock Bit A 0 in this bit indicates that a remote commander has locked access to the MC-MXI from other local sources (front panel switches, front panel serial ports, and so on). This bit is set on a hard reset or soft reset.
6–5r	DL[1–0]	Data Length Bits The MC-MXI uses these two bits to dynamically indicate Word Serial Protocol data length. The value of this read-only field indicates the length of data in the Data (In) registers—Data Low (In), Data High (In), and Data Extended (In)—as shown in the following table. Writing to this bit has no effect.

Bit **Mnemonic** **Description**

DL1	DL0	Data Length
0	0	16 bits
0	1	32 bits
1	1	48 bits

DL1 is set on a write to the low byte (D[7–0]) of the Data Extended (In) register (address offset 0Bh). DL0 is set on a write to the low byte (D[7–0]) of the Data High (In) register (address offset 0Dh). Both bits are cleared on a read of the low byte (D[7–0]) of the Data Low (In) register (address offset 2Fh). These bits are cleared on a hard reset or soft reset.

4r MQEDET Multiple Query Error Detect Bit

The value of this read-only bit is the value of the RDRDY bit on the last falling edge of the WRRDY bit. Under normal situations, the local CPU sets WRRDY. A remote commander clears WRRDY by writing to the Data Low (In) register of the MC-MXI. The local CPU can also use this bit to detect multiple Word Serial query errors. For example, a remote commander might write a data/command value to the Data Low (In) register of the MC-MXI while the local CPU is sending a response to the remote commander. MQEDET is implemented as shown in the following truth table:

RDRDY	WRRDY	MQEDET
0	1 -> 0	0
1	1 -> 0	1
0 -> 1	1 -> 0	1
1 -> 0	1 -> 0	0

The last two entries in the table are shown for cases in which the local CPU changes the value of the RDRDY bit (using a write cycle) at the same time that it is clearing the WRRDY bit (in the same write cycle) from 1 to 0. MQEDET is cleared on a hard reset or a soft reset. Writing to this bit has no effect.

Bit	Mnemonic	Description
3r	RDERR	Read Error Bit A 1 in this bit indicates that a remote commander read one or more of the Data (Out) registers of the MC-MXI when the MC-MXI was not read ready. RDERR is set upon reading any of the Data (Out) registers while the RDRDY (Read Ready) bit in the Response register (bit 10) is 0. Writing a 0 to RDERR clears it. Writing a 1 to RDERR has no effect on its value. This bit is cleared on a hard or soft reset.
2r	WRERR	Write Error Bit A 1 in this bit indicates that a commander wrote to one of the Data (In) registers of the MC-MXI when the MC-MXI was not write ready. WRERR is set upon writing to the lower byte of either the Data Low (In), Data High (In), or Data Extended (In) registers while the WRRDY (Write Ready) bit in the MXI Response register (bit 9) is 0. Writing a 0 to WRERR clears it. Writing a 1 to WRERR has no effect on its value. This bit is cleared on a hard or soft reset.
1–0r	RESP[1–0]	Response Register Bits [1–0] These bits are implemented in the MC-MXI as general-purpose read/write bits. These bits are not affected by a hard or soft reset.

See Also: Data Extended (In) Register, Data High (In) Register, Data Low (In) Register, Data Low (Out) Register.

Data Registers

Function: The Data registers accommodate 16-bit, 32-bit, or 48-bit data and command transfers from a remote commander to the MC-MXI, and they accommodate 16-bit or 32-bit data transfers from the MC-MXI to the remote commander. The data input and output paths are independent, and they are implemented as separate register sets that are accessed according to the table below. The (In) and (Out) notations are relative to the MC-MXI.

Register	Read Offset	Write Offset	Bits	Significance
Data Extended (In)	2Ah	0Ah	47–32	Most Significant
Data High (In)	2Ch	0Ch	31–16	
Data Low (In)	2Eh	0Eh	15–0	Least Significant
Data High (Out)	0Ch	2Ch	31–16	Most Significant
Data Low (Out)	0Eh	2Eh	15–0	Least Significant

The MC-MXI supports the Word Serial Protocol (Normal Transfer mode) defined by the VXIbus specification. The contents of the Data registers are not affected by a hard reset or the RESET bit in the MXI Control register.

See Also: MXI Control Register.

Data Extended (In)

Address:
Micro Channel Read: I/O base address + 2Ah
Micro Channel Write: I/O base address + Ah
MXIbus A16 Space Read: C000h + (LA * 40h) + 2Ah
MXIbus A16 Space Write: C000h + (LA * 40h) + Ah

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
D[47]	D[46]	D[45]	D[44]	D[43]	D[42]	D[41]	D[40]	1
7	6	5	4	3	2	1	0	
D[39]	D[38]	D[37]	D[36]	D[35]	D[34]	D[33]	D[32]	0

Data High (In)

Address:	Micro Channel Read:	I/O base address + 2Ch
	Micro Channel Write:	I/O base address + Ch
	MXIbus A16 Space Read:	C000h + (LA * 40h) + 2Ch
	MXIbus A16 Space Write:	C000h + (LA * 40h) + Ch

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
D[31]	D[30]	D[29]	D[28]	D[27]	D[26]	D[25]	D[24]	1

7	6	5	4	3	2	1	0	
D[23]	D[22]	D[21]	D[20]	D[19]	D[18]	D[17]	D[16]	0

Data Low (In)

Address:	Micro Channel Read:	I/O base address + 2Eh
	Micro Channel Write:	I/O base address + Eh
	MXIbus A16 Space Read:	C000h + (LA * 40h) + 2Eh
	MXIbus A16 Space Write:	C000h + (LA * 40h) + Eh

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
D[15]	D[14]	D[13]	D[12]	D[11]	D[10]	D[9]	D[8]	1

7	6	5	4	3	2	1	0	
D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]	0

Data High (Out)

Address:	Micro Channel Read:	I/O base address + Ch
	Micro Channel Write:	I/O base address + 2Ch
	MXIbus A16 Space Read:	C000h + (LA * 40h) + Ch
	MXIbus A16 Space Write:	C000h + (LA * 40h) + 2Ch

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
D[31]	D[30]	D[29]	D[28]	D[27]	D[26]	D[25]	D[24]	1

7	6	5	4	3	2	1	0	
D[23]	D[22]	D[21]	D[20]	D[19]	D[18]	D[17]	D[16]	0

Data Low (Out)

Address:	Micro Channel Read:	I/O base address + Eh
	Micro Channel Write:	I/O base address + 2Eh
	MXIbus A16 Space Read:	C000h + (LA * 40h) + Eh
	MXIbus A16 Space Write:	C000h + (LA * 40h) + 2Eh

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
D[15]	D[14]	D[13]	D[12]	D[11]	D[10]	D[9]	D[8]	1

7	6	5	4	3	2	1	0	
D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]	0

A24 Pointer Register

Function: Two 16-bit registers implement the A24 Pointer register. The A24 Pointer (High) register stores the most significant 8 bits of the A24 pointer, and the A24 Pointer (Low) register stores the least significant 16 bits. Shared Memory Protocol uses the A24 Pointer register. The contents of the A24 Pointer register are not affected by a hard or soft reset.

A24 Pointer (High)

Address:	Micro Channel Read:	I/O base address + 10h
	Micro Channel Write:	I/O base address + 10h
	MXIbus A16 Space Read:	C000h + (LA * 40h) + 10h
	MXIbus A16 Space Write:	C000h + (LA * 40h) + 10h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
X	X	X	X	X	X	X	X	1

7	6	5	4	3	2	1	0	
A[23]	A[22]	A[21]	A[20]	A[19]	A[18]	A[17]	A[16]	0

Note: ***Bits 15 to 8 of this register do not affect the operation of the MC-MXI. However, in accordance with the VXIbus specification, you can use these bits to read, write, and store data.***

A24 Pointer (Low)

Address:	Micro Channel Read:	I/O base address + 12h
	Micro Channel Write:	I/O base address + 12h
	MXIbus A16 Space Read:	C000h + (LA * 40h) + 12h
	MXIbus A16 Space Write:	C000h + (LA * 40h) + 12h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
A[15]	A[14]	A[13]	A[12]	A[11]	A[10]	A[9]	A[8]	1

7	6	5	4	3	2	1	0	
A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	0

A32 Pointer Register

Function: Two 16-bit registers implement the A32 Pointer register. The A32 Pointer (High) register stores the most significant 16 bits of the A32 pointer, and the A32 Pointer (Low) register stores the least significant 16 bits. Shared Memory Protocol uses the A32 Pointer register. The contents of the A32 Pointer register are not affected by a hard or soft reset.

A32 Pointer (High)

Address:
- Micro Channel Read: I/O base address + 14h
- Micro Channel Write: I/O base address + 14h
- MXIbus A16 Space Read: C000h + (LA * 40h) + 14h
- MXIbus A16 Space Write: C000h + (LA * 40h) + 14h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
A[31]	A[30]	A[29]	A[28]	A[27]	A[26]	A[25]	A[24]	1
7	6	5	4	3	2	1	0	
A[23]	A[22]	A[21]	A[20]	A[19]	A[18]	A[17]	A[16]	0

A32 Pointer (Low)

Address:
- Micro Channel Read: I/O base address + 16h
- Micro Channel Write: I/O base address + 16h
- MXIbus A16 Space Read: C000h + (LA * 40h) + 16h
- MXIbus A16 Space Write: C000h + (LA * 40h) + 16h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
A[15]	A[14]	A[13]	A[12]	A[11]	A[10]	A[9]	A[8]	1
7	6	5	4	3	2	1	0	
A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	0

Soft Reset Service Register

Function: The Soft Reset Service register is a read-only register. Reading this register clears the SRSTREQ bit in the Communication Status register. The value returned is meaningless.

Address:
Micro Channel Read: I/O base address + 24h
MXIbus A16 Space Read: C000h + (LA * 40h) + 24h

Type: Read Only

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
X	X	X	X	X	X	X	X	1

7	6	5	4	3	2	1	0	
X	X	X	X	X	X	X	X	0

See Also: Communication Status Register.

Communication Status Register

Function: The Communication Status register reflects the states of the interrupt enables, the Soft Reset enable, the Logical Address register enable, and the Read, Write, and Soft Reset requests.

Address: Micro Channel Read: I/O base address + 26h
MXIbus A16 Space Read: C000h + (LA * 40h) + 26h

Type: Read Only

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
SIGIERD	RDIERD	WRIERD	SRSTIERD	SRSTENRD	LARENRD	X	X	1

7	6	5	4	3	2	1	0	
SIGREQ	RDREQ	WRREQ	SRSTREQ	X	X	X	X	0

Bit	Mnemonic	Description
15r	SIGIERD	Signal Interrupt Enable Bit (read only) This bit reflects the state of the SIGIE bit in the Communication Control register.
14r	RDIERD	Read Data Interrupt Enable Bit (read only) This bit reflects the state of the RDIE bit in the Communication Control register.
13r	WRIERD	Write Data Interrupt Enable Bit (read only) This bit reflects the state of the WRIE bit in the Communication Control register.
12r	SRSTIERD	Soft Reset Interrupt Enable Bit (read only) This bit reflects the state of the SRSTIE bit in the Communication Control register.
11r	SRSTENRD	Soft Reset Enable Bit (read only) This bit reflects the state of the SRSTEN bit in the Communication Control register.

Bit	Mnemonic	Description
10r	LARENRD	Logical Address Register Enable Bit (read only) This bit reflects the state of the LAREN bit in the Communication Control register.
9–8	X	Unused Bits These bits are reserved for future use and return a meaningless value.
7r	SIGREQ	Signal Request Bit A 1 indicates that one or more values are in the Signal register. A 0 indicates that the Signal register is empty. SIGREQ is logically ANDed with the SIGIE bit to generate an interrupt request on the Micro Channel.
6r	RDREQ	Read Data Request Bit A 1 in this bit indicates that the MC-MXI Data (Out) registers are empty. RDREQ is the logical NOT of the RDRDY bit in the MXI Response register.
5r	WRREQ	Write Data Request Bit A 1 in this bit indicates that a remote commander has placed a data byte into the low byte (D[7–0]) of the MC-MXI Data Low (In) register. WRREQ is cleared when the local CPU reads D[7–0]. WRREQ is logically ANDed with the WRIE bit to generate an interrupt request on the Micro Channel. This bit is cleared on a hard reset or if the RESET bit in the MXI Control register and the SRSTEN bit in the Communication Control register are both set.
4r	SRSTREQ	Soft Reset Request Bit This bit is set when the RESET bit in the MXI Control register changes from a 0 to a 1. This bit is cleared when the Soft Reset Service register is read. SRSTREQ is logically ANDed with the SRSTIE bit to generate an interrupt request on the Micro Channel. This bit is cleared on a hard reset.
3–0	X	Unused Bits These bits are reserved for future use and return a meaningless value.

See Also: Communication Control Register, Data (Out) Registers, Logical Address Register, MXI Control Register, MXI Response Register, MXI Signal Register, Soft Reset Service Register.

Communication Control Register

Function: The Communication Control register controls the different conditions that can trigger a communication interrupt and enables the Soft Reset and the Logical Address register.

Address: Micro Channel Write: I/O base address + 26h
MXIbus A16 Space Write: C000h + (LA * 40h) + 26h

Type: Write Only

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
SIGIEWR	RDIEWR	WRIEWR	SRSTIEWR	SRSTENWR	LARENWR	1	X	1

7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0

Bit	Mnemonic	Description
15w	SIGIEWR	Signal Interrupt Enable Bit (write only) Setting this bit enables signal interrupts. Clearing this bit disables the interrupt. The signal interrupt triggers if the Signal register is not empty. A hard or soft reset clears this bit.
14w	RDIEWR	Read Data Interrupt Enable Bit (write only) Setting this bit enables Read Data interrupts. Clearing this bit disables interrupt generation. This interrupt triggers when the RDREQ bit in the Communication Status register is 1. A hard or soft reset clears this bit.
13w	WRIEWR	Write Data Interrupt Enable Bit (write only) Setting this bit enables Write Data interrupts. Clearing this bit disables the interrupt. This interrupt triggers when the WRREQ bit in the Communication Status register is 1. A hard or soft reset clears this bit.
12w	SRSTIEWR	Soft Reset Interrupt Enable Bit (write only) Setting this bit enables Soft Reset interrupts. Clearing this bit disables interrupt generation. This interrupt triggers on a soft reset. A hard reset clears this bit.

Bit	Mnemonic	Description
11w	SRSTENWR	Soft Reset Enable Bit (write only) Setting this bit enables the Soft Reset function. If SRSTEN is set, a Soft Reset occurs when the RESET bit in the MXI Control register is set. A hard reset clears this bit.
10w	LARENWR	Logical Address Register Enable Bit (write only) Setting this bit enables the MXI Logical Address decode logic. Clearing this bit disables the decode logic. A hard reset clears this bit.
9–0	1000000000	Reserved Bits These bits are reserved for future use and should be written with the value 1000000000 (binary). A hard reset clears these bits.

See Also: Communication Status Register, Logical Address Register, MXI Control Register, MXI Signal Register.

Slave Configuration Register

Function: The slave configuration register controls several micro-channel specific features that are applied to MXI slave-made transfers.

Address:

Micro Channel Read:	I/O base address + 30h
Micro Channel Write:	I/O base address + 30h
MXIbus A16 Space Read:	C000h + (LA * 40h) + 30h
MXIbus A16 Space Write:	C000h + (LA * 40h) + 30h

Type: Read/Write

Size: 8-bit or 16-bit accessible

Bit Map:

15	14	13	12	11	10	9	8	
X	X	X	0	0	0	0	0	1

7	6	5	4	3	2	1	0	
X	X	X	LOCKMC	SWAPEN	INTACK	M/IO*	X	0

Bit	Mnemonic	Description
15–13	X	Unused bits Reading these bits returns a meaningless value.
12–8	0	Reserved These bits should always be written with 0s.
7–5	X	Unused bits Reading these bits returns a meaningless value.
4r/w	LOCKMC	Lock Micro Channel Setting this bit to 0 forces the MC-MXI to arbitrate for the Micro Channel and to maintain its hold over the Micro Channel after arbitration is successful. If a local timeout forces the MC-MXI to yield the Micro Channel, the MC-MXI immediately begins to arbitrate for the Micro Channel again.

Bit	Mnemonic	Description
3r/w	SWAPEN	Enable Byte Swapping Setting this bit instructs the MC-MXI to swap the byte ordering of data passing from the MXIbus to the Micro Channel. Clearing this bit permits data to pass without swapping. This bit is effective only during MXIbus slave-mode operations. To control byte swapping during MXIbus master-mode transfers, use the SWAPMB bit in the MC-MXI–CTRL register.
2r/w	INTACK	Acknowledge Bit This bit serves as a read/write bit for handshaking between the local CPU and a remote device on the MXIbus. It does not affect any MC-MXI operations.
1r/w	M/IO*	Memory/IO select Clearing this bit causes MXIbus slave-mode data transfers to map the Micro Channel I/O address space. Setting M/IO* maps MXIbus slave-mode transfers to the Micro Channel memory address space.
0	X	Unused bit This bit is unused and returns a meaningless value.

See Also: MC-MXI–CTRL Register.

MC-MXI Local Registers

PITLOCAL Register

Description: Local bus timeout

Function: The PITLOCAL register specifies the amount of time before a Micro Channel timeout is generated during MXI slave-mode operations.

Address: Micro Channel Read: I/O base address + 48h
Micro Channel Write: I/O base address + 48h

Type: Read/Write

Size: 8-bit accessible

7	6	5	4	3	2	1	0
T[15]/T[7]	T[14]/T[6]	T[13]/T[5]	T[12]/T[4]	T[11]/T[3]	T[10]/T[2]	T[9]/T[1]	T[8]/T[0]

Bit	Mnemonic	Description
7–0r/w	T[15–0]	Time Value bits This register specifies the time from the beginning of a MXIbus slave-mode operation to the generation of a Micro Channel timeout signal. This timeout signal precludes the MC-MXI from tying up the Micro Channel beyond the limits defined by the Micro Channel specification. The specified time duration is a 16-bit number written to the PITLOCAL register in the order of the least significant byte followed by the most significant byte. The time duration of the timeout clock is specified in increments of 0.1 µs. When a MXIbus slave operation begins, the contents of the register are copied into a 16-bit counter which, in turn, decrements every 0.1 µs. If the counter reaches 0 and the operation has not completed, a timeout signal generates. The counter reloads every time a new MXIbus slave operation occurs. The maximum recommended timeout value for PITLOCAL is 7.6 µs, which translates into a count of 004Ch.

It is possible to read the current count in the PITLOCAL register. To do this, first write 00h to the PITCNTR register. Performing this action latches the contents of the PITLOCAL 16-bit counter. The normal operation of the PITLOCAL timer is not affected by this operation. Next, perform two read operations to the PITLOCAL register. The first read returns the least significant byte and the second read returns the most significant byte.

Note: ***Reading the PITLOCAL register without writing 00h to the PITCNTR register could result in an undefined value.***

See Also: PITCNTR Register.

PITREADY Register

Description: Local ready bus timeout

Function: The PITREADY register specifies the amount of time before a Micro Channel timeout is generated during MXI master-mode operations.

Address: Micro Channel Read: I/O base address + 4Ah
Micro Channel Write: I/O base address + 4Ah

Type: Read/Write

Size: 8-bit accessible

7	6	5	4	3	2	1	0
T[15]/D[7]	T[14]/T[6]	T[13]/T[5]	T[12]/T[4]	T[11]/T[3]	T[10]/T[2]	T[9]/T[1]	T[8]/T[0]

Bit	Mnemonic	Description
7–0r/w	T[15–0]	Time Value bits This register specifies the time from the beginning of a MXIbus master-mode operation to the generation of a Micro Channel timeout signal. This timeout signal precludes the MC-MXI from tying up the Micro Channel beyond the limits defined by the Micro Channel specification. The specified time duration is a 16-bit number that is written to the PITREADY register in the order of the least significant byte followed by the most significant byte. The time duration of the timeout clock is specified in increments of 0.1 µs. When a MXIbus master operation begins, the contents of the register are copied into a 16-bit counter, which in turn decrements every 0.1 µs. If the counter reaches 0 and the operation has not completed, a timeout signal is generated. The counter reloads every time a new MXIbus master operation occurs. The recommended time from initiating a MXI master-mode operation to generating a timeout signal is 3.5 µs. This time value corresponds to a PITREADY count of 0023h.

It is possible to read the current count in the PITREADY register. To do this, first write 40h to the PITCNTR register. Performing this action latches the contents of the PITREADY 16-bit counter. The normal operation of the PITREADY timer is not affected by this operation. Next, perform two read operations to the PITREADY register. The first read returns the least significant byte and the second read returns the most significant byte.

Note: ***Reading the PITREADY register without writing 40h to the PITCNTR register could result in an undefined value.***

See Also: PITCNTR Register.

PITSC Register

Description: MXIbus System Controller Timeout

Function: The PITSC register specifies the amount of time from the initiation of any MXIbus transaction to the generation of a System Controller timeout.

Address: Micro Channel Read: I/O base address + 4Ch
Micro Channel Write: I/O base address + 4Ch

Type: Read/Write

Size: 8-bit accessible

7	6	5	4	3	2	1	0
T[15]/D[7]	T[14]/T[6]	T[13]/T[5]	T[12]/T[4]	T[11]/T[3]	T[10]/T[2]	T[9]/T[1]	T[8]/T[0]

Bit **Mnemonic** **Description**

7–0r/w T[15–0] Time Value bits

This register specifies the time from the beginning of any MXIbus operation to the generation of a System Controller timeout signal on the MXIbus. This timer is used if the MC-MXI is configured as a MXIbus System Controller.

The specified time duration is a 16-bit number that is written to the PITREADY register in the order of the least significant byte followed by the most significant byte. The time duration of the timeout clock is specified in increments of 0.8 µs (1.25 MHz).

When a MXIbus operation begins, the contents of the register are copied into a 16-bit counter, which in turn decrements every 0.8 µs. If the counter reaches 0 and the operation has not completed, a System Controller timeout signal is generated. The counter reloads every time a new MXIbus operation occurs.

The recommended time from initiating a MXIbus operation to generating a System Controller timeout signal is 1 ms. This time value corresponds to a PITSC count of 04E2h.

It is possible to read the current count in the PITSC register. To do this, first write 80h to the PITCNTR register. Performing this action latches the contents of the PITSC 16-bit counter. The normal operation of the PITSC timer is not affected by this operation. Next, perform two read operations to the PITSC register. The first read returns the least significant byte and the second read returns the most significant byte.

Note: ***Reading the PITSC register without writing 80h to the PITCNTR register could result in an undefined value.***

See Also: PITCNTR Register.

PITCNTR Register

Description: Control register for PITLOCAL, PITREADY, and PITSC timers

Function: The PITCNTR register configures and initializes the timers controlled by the PITLOCAL, PITREAD, and PITSC registers.

Address: Micro Channel Write: I/O base address + 4Eh

Type: Write Only

Size: 8-bit accessible

7	6	5	4	3	2	1	0
SC[1]	SC[0]	BRW[1]	BRW[0]	M[2]	M[1]	M[0]	BCD

Warning: ***The PITCNTR, PITLOCAL, PITREADY, and PITSC registers are part of the Intel 82C54 timer component. Although this component has many different modes of operation, use only the modes described in this manual. Attempting other modes could result in improper operation of and/or damage to the MC-MXI.***

Bit **Mnemonic** **Description**

7–6w SC[1–0] Select Clock

These bits specify which of the PITLOCAL, PITREADY, and PITSC timers will be configured. The following table shows the selection of the registers.

SC[1]	SC[0]	Action
0	0	Select PITLOCAL
0	1	Select PITREADY
1	0	Select PITSC
1	1	Not used by MC-MXI

5–4w BRW[1–0] Byte order for timer read/write

Sets the byte ordering on read and write operations to the PITLOCAL, PITREADY, and PITSC registers. Always set these bits to 11b.

Bit	Mnemonic	Description
3–1w	M[2–0]	Operating Mode bits Specifies the operating mode for the PITLOCAL, PITREADY, and PITSC timers. Always set these bits to 010b.
0w	BCD	Binary or Decimal Counting bit This bit dictates the counting mode of the PITLOCAL, PITREADY, and PITSC timers. Because the MC-MXI always uses binary counting, you should always set this bit to 0.

Note: ***To properly initialize the timers, write the following three bytes consecutively to the PITCNTR.***

Byte	Action
34h	Configures the PITLOCAL timer
74h	Configures the PITREADY timer
b4h	Configures the PITSC timer

Along with the latching commands used to read the values of the PITLOCAL, PITREADY, and PITSC registers, these three bytes are the only values you should *ever* write to the PITCNTR register.

See Also: PITLOCAL Register, PITREADY Register, PITSC Register.

WIN1AM-SMOFF Register

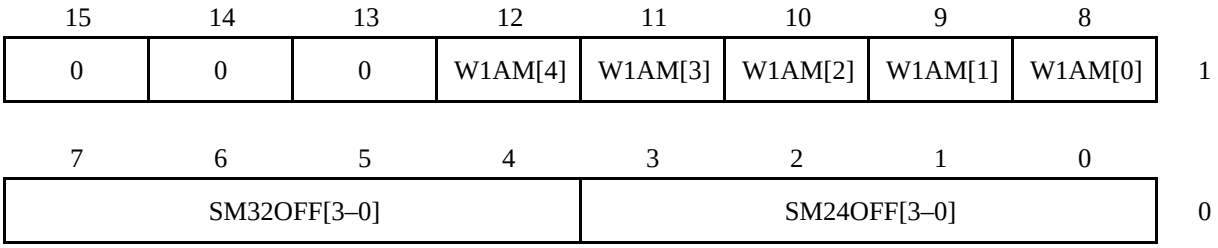
Description: Master-mode address modifier register for window 1 and slave-mode offset register

Function: The most significant byte of this register defines the address modifier code that is driven on the MXIbus during accesses through window 1. The least significant byte provides a memory offset register that remaps MXIbus addresses to Micro Channel addresses during slave-mode transfers from the MXIbus to the Micro Channel.

Address: Micro Channel Write: I/O base address + 50h

Type: Write Only

Size: 8-bit or 16-bit accessible



Bit	Mnemonic	Description
15–13	0	Reserved Bits These bits are reserved for future use. Always write 0s to these bits.
12–8w	W1AM[4–0]	Window 1 Address Modifier Bits These bits are driven onto MXIbus address modifier lines AM[4–0]* during master-mode MXIbus transfers through memory window 1.
7–4w	SM32OFF[3–0]	Slave Mode A32 Offset Bits The value of these bits is added to MXIbus address lines AD[31–28]* to form the Micro Channel address used during slave-mode A32 accesses to Micro Channel memory. This method is used to remap physical MXIbus address space to a different Micro Channel address space. The 4-bit composition of this offset block makes it possible to remap Micro Channel memory on 256-MB boundaries during A32 slave-mode transfers.

Bit	Mnemonic	Description
		The value of these bits should be considered a 2's complement number when determining the mapping function. The MC-MXI discards any carry bits when adding the offset bits to the incoming MXIbus address. As a result, Micro Channel addresses can be either higher or lower than the MXIbus address. These bits have no effect during slave-mode A24 accesses to Micro Channel memory.
3–0w	SM24OFF[3–0]	Slave Mode A24 Offset Bits The value of these bits is added to MXIbus address lines AD[23–19] to form the Micro Channel address used during slave-mode A24 accesses to Micro Channel memory. This method is used to remap physical MXIbus address space to a different Micro Channel address space. The 4-bit composition of this offset block makes it possible to remap Micro Channel memory on 1-MB boundaries during A24 slave-mode transfers. The value of these bits should be considered a 2's complement number when determining the mapping function. The MC-MXI discards any carry bits when adding the offset bits to the incoming MXIbus address. As a result, Micro Channel addresses can be either higher or lower than the MXIbus address. These bits should be programmed with 0s if the MC-MXI is configured for A32 slave-mode operation.

WIN3AM-WIN2AM Register

Description: Master-mode address modifier registers for windows 2 and 3

Function: These registers define the address modifier code that is driven on the MXIbus during accesses through windows 2 or 3.

Address: Micro Channel Write: I/O base address + 58h

Type: Write Only

Size: 8-bit or 16-bit accessible

15	14	13	12	11	10	9	8	
0	0	0	W3AM[4]	W3AM[3]	W3AM[2]	W3AM[1]	W3AM[0]	1

7	6	5	4	3	2	1	0	
0	0	0	W2AM[4]	W2AM[3]	W2AM[2]	W2AM[1]	W2AM[0]	0

Bit	Mnemonic	Description
15–13	0	Reserved Bits These bits are reserved for future use. Always write 0s to these bits.
12–8w	W3AM[4–0]	Window 3 Address Modifier Bits These bits are driven onto MXIbus address modifier lines AM[4–0] during master-mode MXIbus transfers through memory window 3.
7–5	0	Reserved Bits These bits are reserved for future use. Always write 0s to these bits.
4–0w	W2AM[4–0]	Window 2 Address Modifier Bits These bits are driven onto MXIbus address modifier lines AM[4–0] during master-mode MXIbus transfers through memory window 2.

Master Mode Page Register

Description: Master-mode address page register

Function: This register provides the upper order address lines during MXIbus master-mode transfers.

Address: Micro Channel Read: I/O base address + 60h
Micro Channel Write: I/O base address + 60h

Type: Read/Write

Size: Write: 8 bit and 16 bit

15	14	13	12	11	10	9	8	
A[31]	A[30]	A[29]	A[28]	A[27]	A[26]	A[25]	A[24]	1
7	6	5	4	3	2	1	0	
A[23]	A[22]	A[21]	A[20]	A[19]	A[18]	A[17]	A[16]	0

Bit	Mnemonic	Description
15–8r/w	A[31–24]	Address Bits for Lines 31 through 24 These bits are driven onto MXIbus address lines AD[31–24] during master-mode MXIbus transfers through any of the three memory windows. These bits need to be programmed only prior to master-mode A32 transfers because A24 and A16 slaves do not use address lines AD[31–24].
7–0r/w	A[23–16]	Address Bits for Lines 23 through 16 These bits are driven onto MXIbus address lines AD[23–16] during master-mode MXIbus transfers through memory window 1. Memory window 1 is only 64 KB in size. Accesses through this window supply only the low-order 16 address lines to the MXIbus. Therefore, this register must be programmed prior to attempting master-mode A24 or A32 transfers through window 1. During MXIbus master-mode transfers through memory windows 2 or 3, MXIbus address lines AD[23–16] are driven directly from corresponding Micro Channel address lines.

MC-MXI–STATUS Register

Description: MC-MXI board status register

Function: This register provides status information on the operation of the MC-MXI and the MXIbus.

Address: Micro Channel Read: I/O base address + 68h

Type: Read Only

Size: 8-bit or 16-bit accessible

15	14	13	12	11	10	9	8	
1	1	1	1	PERR	DL	BERR	TO	1
7	6	5	4	3	2	1	0	
PCREQINT	OWNMBINT	TCINT	RMBIRQ	PCREQ	MBREQ	BLOCK	OWNMB	0

Bit	Mnemonic	Description
15–12	1	Reserved Bits These bits are not used and always return a 1.
11r	PERR	Parity Error Bit This bit returns a 1 if a parity error occurred on the last master-mode MXIbus transfer. No assumptions can be made as to whether the transfer actually took place. This bit is cleared when it is read.
10r	DL	Deadlock Bit This bit returns a 1 if the last master-mode MXIbus transfer terminated prematurely because of a deadlock condition. If set, this bit indicates that the MXIbus transfer did not take place and should be retried. This bit is cleared when it is read.
9r	BERR	Bus Error Bit This bit returns a 1 if the last master-mode MXIbus transfer terminated with a bus error. No assumptions can be made as to either the cause of the bus error or whether the transfer actually took place. This bit is cleared when it is read.

Bit	Mnemonic	Description
8r	TO	Timeout Bit This bit returns a 1 if the last master-mode MXIbus transfer terminated prematurely because of a Micro Channel timeout. This is not an error condition: it is merely an indication that the transfer could not complete within the maximum-allowed Micro Channel access time. If this bit is set, the MXIbus transfer is still ongoing and the previous Micro Channel master-mode transfer should immediately be re-executed until it either completes successfully or terminates with a bus error condition.
7r	PCREQINT	PC Request Interrupt Bit This bit returns a 1 if PC (Micro Channel) resources have been requested by a remote MXIbus device via a slave-mode MXIbus transfer and the PCREQIE bit in the Board Control register is set. This bit can be used to alert the CPU that an external device is trying to access local resources. Writing a 0 to PCREQIE or generating a hard reset clears PCREQINT.
6r	OWNMBINT	MXIbus Ownership Interrupt Bit This bit returns a 1 when the MC-MXI wins control of the MXIbus as a master and the OWNMBIE bit in the Control register is set. This bit can be used to alert the Micro Channel bus master that it has won control of the MXIbus through arbitration. Writing a 0 to the OWNMBIE bit or generating a hard reset clears OWNMBINT.
5r	TCINT	Terminal Count Interrupt Bit The MC-MXI sets this bit to 1 to indicate that a DMA controller on the Micro Channel has finished transferring data to the MC-MXI. The MC-MXI sets this bit when the DMA controller asserts the Micro Channel *TC* signal and the TCIE bit in the MC-MXI–CTRL register is set. You can use this bit to alert the local CPU that the DMA operation to the MC-MXI is complete. Writing a 0 to the TCIE bit or generating a hard reset clears TCINT.
4r	RMBIRQ	MXIbus IRQ* Bit This bit returns a 1 when the MXIbus IRQ* line is asserted. It returns a 0 when the MXIbus IRQ* line is unasserted.
3r	PCREQ	PC Request Bit This bit returns a 1 when an external MXIbus master requests PC (Micro Channel) resources. It returns a 0 when no requests for Micro Channel resources are pending.

Bit	Mnemonic	Description
2r	MBREQ	MXIbus Request Bit This bit returns a 1 when the MC-MXI requests access to the MXIbus. It returns a 0 when the MC-MXI is not requesting the MXIbus.
1r	BLOCK	Block Mode Master Transfer Bit This bit returns a 1 if the MC-MXI is programmed for a block-mode MXIbus master transfer that it has not yet completed. It returns a 0 when no block-mode master transfers are pending.
0r	OWNMB	MXIbus Ownership Bit This bit returns a 1 when the MC-MXI owns control of the MXIbus. It returns a 0 when the MC-MXI does not own control of the MXIbus.

See Also: MC-MXI–CTRL Register.

MC-MXI–CTRL Register

Description: MC-MXI board control register

Function: This register controls MC-MXI functions such as enabling interrupts and error condition reporting.

Address: Micro Channel Write: I/O base address + 68h

Type: Write Only

Size: 8-bit or 16-bit accessible

15	14	13	12	11	10	9	8	
MBSC	LOCKMB	FAIRMB	SWAPMB	CHCKEN	MMEN	DMBIRQ	0	1

7	6	5	4	3	2	1	0	
PCREQIE	OWNMBIE	TCIE	TCENDEN	MBBLOCKEN	MDRQEN	0	0	0

Bit	Mnemonic	Description
15w	MBSC	MXIbus System Controller Bit Setting this bit configures the MC-MXI to be the MXIbus System Controller. If this bit is cleared, the MC-MXI will not be the MXIbus System Controller and another device on the MXIbus must accept this responsibility. This bit is cleared on a hard reset.
14w	LOCKMB	Lock MXIbus Bit Setting this bit locks the MXIbus during master-mode transfers, so no other MXIbus master can interfere with the master-mode operations of the MC-MXI. The MC-MXI locks the MXIbus by asserting the MXIbus *BUSY** signal, thereby precluding other masters from arbitrating for the bus. When this bit is cleared, other masters can arbitrate for the MXIbus. This bit is cleared on a hard reset.
13w	FAIRMB	Fair MXIbus Requester Bit Setting this bit makes the MC-MXI a fair MXIbus requester. As a fair MXIbus device, the MC-MXI does not request the MXIbus until it detects that the MXIbus *BREQ** signal is unasserted. When this bit is cleared, the MC-MXI can request the bus at will. A hard reset clears this bit.

Bit	Mnemonic	Description
12w	SWAPMB	Multi-Byte Swap Bit Setting this bit causes all multi-byte MXIbus master-mode transfers to swap the position of the bytes within words and words within longwords. When this bit is cleared, data is transferred without alteration. This bit allows processors of different types to share resources without being required to swap data types and formats in software. A hard reset clears this bit.
11w	CHCKEN	I/O Channel Check Interrupt Enable Bit Setting this bit enables master-mode error conditions to drive the Micro Channel channel check (CHCK) signal. A hard reset clears this bit.
10w	MMEN	Master Mode Enable Bit When this bit is set, the MC-MXI can perform master-mode MXIbus transfers. Clearing this bit disables MC-MXI master-mode operation and resets the master-mode state machine. A hard reset clears this bit.
9w	DMBIRQ	Drive MXIbus IRQ* Bit Writing a 1 to this bit causes the MC-MXI to assert the IRQ* line on the MXIbus. Writing a 0 to this bit unasserts the MXIbus IRQ* line. This bit is cleared on a hard reset.
8	0	Reserved Bits These bits are reserved for future use. Always write 0s to these bits.
7w	PCREQIE	PC Request Interrupt Enable Bit Writing a 1 to this bit enables the *PCREQINT* signal. A PC request interrupt triggers when a MXIbus device attempts to access the Micro Channel. Clearing this bit disables the *PCREQINT* signal status and resets the PCREQINT condition. A hard reset clears PCREQIE.
6w	OWNMBIE	MXIbus Ownership Interrupt Enable Bit Writing a 1 to this bit enables the *OWNMBINT* signal. Clearing this bit disables the *OWNMBINT* signal status and resets the OWNMBINT condition. A hard reset clears OWNMBIE.

Bit	Mnemonic	Description
5w	TCIE	Terminal Count Interrupt Enable Bit Writing a 1 to this bit enables the *TCINT* signal. Clearing this bit disables the *TCINT* signal status and resets the TCINT condition. A hard reset clears TCIE.
4w	TCENDEN	Terminal Count Enable Bit Setting this bit enables a TC (terminal count) condition to terminate a MXIbus block-mode operation automatically. When this bit is cleared, a terminal count condition has no effect on the ongoing MXIbus block-mode operation. A hard reset clears TCENDEN.
3w	MBBLOCKEN	MXIbus Block Mode Enable Bit Writing a 1 to this bit causes the next MC-MXI master-mode MXIbus transfer to start a MXIbus block-mode operation. This bit should be cleared prior to the last transfer in the block-mode operation. This bit is cleared automatically during DMA transfers when the TCENDEN bit is set. A hard reset clears MBBLOCKEN.
2w	MDRQEN	Master Mode DMA Transfer Enable Bit Writing a 1 to this bit enables master-mode DMA transfers. All DMA transfers use a MXIbus block-mode data transfer. Clearing this bit disables master-mode DMA operations. A hard reset clears MDRQEN.
1–0	0	Reserved Bits These bits are reserved for future use. Always write 0s to these bits.

WAITSTATES Register

Description: Memory and I/O wait states generator control

Function: This register specifies the number of wait states the MC-MXI should use when performing data transfers.

Address: Micro Channel Read: I/O base address + 140h
Micro Channel Write: I/O base address + 140h

Type: Read/Write

Size: 8-bit accessible

7	6	5	4	3	2	1	0
HRDRESET	SYSCLKEN	IOW[2]	IOW[1]	IOW[0]	MW[2]	MW[1]	MW[0]

Bit	Mnemonic	Description
7r/w	HRDRESET	Hard Reset Bit Setting this bit generates a hard reset on the MC-MXI. The contents of the POS registers and the I/O registers from 140h through 145h, which are above the I/O base address, are not affected. All other registers and circuits on the MC-MXI are affected. Clearing this bit removes the hard reset. The hard reset is part of a reset hierarchy that is described in more detail in Chapter 6, *Theory of Operation*.
6r/w	SYSCLKEN	System Controller Timeout Enable Bit Setting this bit to 1 enables the System Controller base clock (SYSCLK). The MC-MXI uses this clock when the MC-MXI is a MXIbus System Controller. Setting this bit enables only SYSCLK. You have to set additional bits in the SYSCLK register to specify the period of the base clock.
5–3r/w	IOW[2–0]	I/O Wait States Bit Field This value determines the number of wait states the MC-MXI uses when performing I/O data transfers (such as registers read/writes). The number of wait states is equal to 7 minus the value of IOW. The MC-MXI normally should be set for two I/O wait states (IOW = 101b).

Bit	Mnemonic	Description
2–0r/w	MW[2–0]	Memory Wait States Bit Field This value determines the number of wait states the MC-MXI uses when performing memory data transfers. The number of wait states is equal to 7 minus the value of MW. The MC-MXI normally should be set for two memory wait states (MW = 101b).

See Also: SCCLK Register.

DMA Enable Register

Description: Setup DMA Channels

Function: This register enables the two DMA channels of the MC-MXI and provides support for burst-mode data transfers.

Address: Micro Channel Read: I/O base address + 141h
Micro Channel Write: I/O base address + 141h

Type: Read/Write

Size: 8-bit accessible

7	6	5	4	3	2	1	0
0	0	BURST2EN	BURST1EN	0	0	DMA2EN	DMA1EN

Bit	Mnemonic	Description
7–6, 3–2	0	Unused bits These bits are not used and should be set to 0.
5r/w	BURST2EN	Burst Enable for DMA Channel 2 Bit When this bit is set to 1, DMA Channel 2 operates in burst mode. During burst mode, the MC-MXI transmits data as a string of uninterrupted data transfers. If burst mode is disabled (that is, BUSRT2EN is set to 0), the MC-MXI releases the Micro Channel and arbitrates before each data transfer. This bit defaults to 0 on a channel reset.
4r/w	BURST1EN	Burst Enable for DMA Channel 1 Bit This bit performs the same function as BURST2EN, but applies to DMA Channel 1.
1r/w	DMA2EN	DMA Channel 2 Enable Bit Setting this bit to 1 enables DMA Channel 2, which controls data transfers from the Micro Channel to the MXIbus. It is set to 0 on a channel reset.
0r/w	DMA1EN	DMA Channel 1 Enable Bit Setting this bit to 1 enables DMA Channel 1, which controls data transfers from the MXIbus to the Micro Channel. It is set to 0 on a channel reset.

Interrupt Enable Register

Description: Enable MC-MXI Interrupts

Function: This register enables the MC-MXI interrupts to the Micro Channel.

Address: Micro Channel Read: I/O base address + 142h
Micro Channel Write: I/O base address + 142h

Type: Read/Write

Size: 8-bit accessible

7	6	5	4	3	2	1	0
0	SYSCLKINTEN	COMMEN	MXIRQEN	MISCEN	MMERREN	0	0

Bit	Mnemonic	Description
7, 1–0	0	Unused bits These bits are not used and should be set to 0.
6r/w	SYSCLKINTEN	System Controller Base Clock Interrupt Enable Setting this bit allows the System Controller base clock to generate interrupts. The MC-MXI does not use the features controlled by this bit, and you must *always set this bit to 0*.
5r/w	COMMEN	Communication Interrupt Enable Bit When this bit is set to 1, the MC-MXI transmits COMM interrupts, which are generated by either message passing or Word Serial Protocols, to the Micro Channel. The Interrupt Map register sets the Micro Channel interrupt priority of this interrupt. If COMMEN is set to 0, the MC-MXI does not transmit the interrupt to the Micro Channel. In either case, if a communication interrupt occurs, the MC-MXI sets the corresponding bit in the MC INT Status register set. A channel reset clears COMMEN.
4r/w	MXIRQEN	MXIbus Interrupt Enable Bit When this bit is set to 1, the MC-MXI transmits interrupts generated on the MXIbus *IRQ** signal to the Micro Channel. The Interrupt Map register sets the Micro Channel interrupt priority of this interrupt. If MXIRQEN is set to 0, the interrupt is not transmitted to the Micro Channel. In either case, if a MXIRQ interrupt occurs, the MC-MXI sets the corresponding bit in the MC INT Status register set. A channel reset clears MXIRQEN.

Bit	Mnemonic	Description
3r/w	MISCEN	General Interrupt Enable Bit When this bit is set to 1, interrupts generated by terminal count (TCINT), MXIbus ownership (OWNMBINT), and requests for Micro Channel accesses (PCREQINT) are transmitted to the Micro Channel. The Interrupt Map register sets the Micro Channel interrupt priority of this interrupt. If MISCEN is set to 0, the interrupt is not transmitted to the Micro Channel. In either case, if a MISCINT interrupt occurs, the MC-MXI sets the corresponding bit in the MC INT Status register set. A channel reset clears MISCEN.
2r/w	MMERREN	Error Condition Interrupt Enable Bit When this bit is set to 1, interrupts generated by error conditions on the MC-MXI are transmitted to the Micro Channel. The Interrupt Map register sets the Micro Channel interrupt priority of this interrupt. If MMERREN is set to 0, the interrupt is not transmitted to the Micro Channel. In either case, if a MMERREN interrupt occurs, the MC-MXI sets the corresponding bit in the MC INT Status register set. A channel reset clears MMERREN.

See Also: Interrupt Map Register.

MC INT Status Register

Description: MC-MXI Interrupt Status Register

Function: This register contains information on the state of different MC-MXI interrupt conditions.

Address: Micro Channel Read: I/O base address + 143h

Type: Read Only

Size: 8-bit accessible

7	6	5	4	3	2	1	0
ANYINT	X	COMMST	MXIRQST	MISCST	MMERRST	0	0

Bit	Mnemonic	Description
7r	ANYINT	Any Interrupt Bit Triggering the TIMER, MIGA, MXIRQ, or MMERR interrupts sets this bit to 1.
6	X	Reserved This bit returns a meaningless value.
5r	COMMST	Communication Interrupt Status Bit If COMMST = 1, a message-passing or Word Serial Protocol transaction triggered an interrupt. This bit is set regardless of the state of the corresponding enable bit in the Interrupt Enable register.
4r	MXIRQST	MXIbus IRQ* Interrupt Status Bit If MXIRQST = 1, the MC-MXI received an interrupt on the MXIbus *IRQ** signal. This bit is set regardless of the state of the corresponding enable bit in the Interrupt Enable register.
3r	MISCST	General Interrupt Status Bit If MISCST = 1, a terminal count (TCINT), MXIbus ownership (OWNMBINT), or request for Micro Channel accesses (PCREQINT) condition generated an interrupt. This bit is set regardless of the state of the corresponding enable bit in the Interrupt Enable register.

Bit	Mnemonic	Description
2r	MMERRST	Error Condition Interrupt Status Bit If this bit = 1, an error condition on the MC-MXI generated an interrupt. This bit is set regardless of the state of the corresponding enable bit in the Interrupt Enable register.
1–0	0	These bits are not used and should be set to 0.

MC INT Ack. Register

Description: MC-MXI Interrupt Acknowledge register

Function: This register acknowledges and resets MC-MXI interrupts.

Address: Micro Channel Write: I/O base address + 143h

Type: Write Only

Size: 8-bit accessible

7	6	5	4	3	2	1	0
TIMERTST	TIMERAK	COMMAK	MXIRQAK	MISCAK	MMERRAK	0	0

Bit	Mnemonic	Description
7w	TIMERTST	Special Mode for System Controller Timeout Bit This bit enables a special mode for the System Controller timer and must *always* be set to 1.
6w	TIMERAK	Timer Interrupt Acknowledge Bit Setting this bit to 1 resets the interrupt status bit TIMERST and prepares it to receive new interrupts. In general, you should set TIMERAK to 0.
5w	COMMAK	Communication Interrupt Acknowledge Bit Writing a 1 to this bit resets the COMMST bit in the MC INT Status register and prepares it to receive new interrupts.
4w	MXIRQAK	Acknowledge Interrupts from the MXIbus *IRQ** Signal Bit Writing a 1 to this bit resets the interrupt status bit MXIRQST and prepares it to receive new interrupts. Setting this bit does not necessarily constitute an acknowledgment of the MXIbus *IRQ** signal. The local CPU may have to perform other operations to properly acknowledge the MXI *IRQ** signal.
3w	MISCAK	General Interrupt Acknowledge Bit Writing a 1 to this bit resets the interrupt status bit MISCST and prepares it to receive new interrupts.
2w	MMERRAK	Error Condition Interrupt Acknowledge Bit Writing a 1 to this bit resets the interrupt status bit MMERRST and prepares it to receive new interrupts.

See Also: MC INT Status Register.

SCCLK Register

Description: System Controller Timer

Function: Sets the base clock period of the System Controller timeout timer.

Address: Micro Channel Read: I/O base address + 144h
Micro Channel Write: I/O base address + 144h

Type: Read/Write

Size: 8-bit accessible

7	6	5	4	3	2	1	0
SCCLK[7]	SCCLK[6]	SCCLK[5]	SCCLK[4]	SCCLK[3]	SCCLK[2]	SCCLK[1]	SCCLK[0]

Bit	Mnemonic	Description
7–0r/w	SCCLK[7–0]	System Controller Clock Period This value specifies the base clock period used to generate the System Controller timeout signal. The System Controller timeout is generated after a predetermined number of clock cycles have elapsed. The number of cycles is specified by the PITSC register. T[7–0] should be written with EFh, which results in a clock frequency of 1.25 MHz. If for some reason you need to change this value, use the following formula to determine the clock frequency: 20 MHz/(16 - T[7–4]) * 8 As the formula implies, the most significant nibble of T[7–0] determines the clock frequency. While it is possible to change the clock frequency, you should set the clock frequency to 1.25 MHz and use the PITSC register to specify System Controller timeouts.

See Also: PITSC Register.

DMA Control Register

Description: Control of DMA Access

Function: This register specifies DMA request triggering between the MC-MXI and the Micro Channel.

Address: Micro Channel Read: I/O base address + 145h
Micro Channel Write: I/O base address + 145h

Type: Read/Write

Size: 8-bit accessible

7	6	5	4	3	2	1	0
AND/OR DMA2	AND/OR DMA1	0	0	0	0	S/W DMA2 REQUEST	S/W DMA1 REQUEST

Bit	Mnemonic	Description
7r/w	AND/OR DMA2	AND or OR Software and Hardware DMA Requests Bit When this bit is set to 1, hardware DMA requests on DMA Channel 2 and the state of S/W DMA2 REQUEST are ANDed together to generate a DMA request on the Micro Channel. If AND/OR DMA2 is set to 0, hardware DMA requests on DMA Channel 2 and the state of S/W DMA2 REQUEST are ORed together to generate a DMA request on the Micro Channel. This bit defaults to 0 on a channel reset.
6r/w	AND/OR DMA1	AND or OR Software and Hardware DMA Requests Bit When this bit is set to 1, hardware DMA requests on Channel 1 and the state of S/W DMA1 REQUEST are ANDed together to generate a DMA request on the Micro Channel. If AND/OR DMA1 is set to 0, hardware DMA requests on Channel 1 and the state of S/W DMA1 REQUEST are ORed together to generate a DMA request on the Micro Channel. This bit defaults to 0 on a channel reset.
5–2	0	Reserved These bits should always be set to 0.

Bit	Mnemonic	Description
1r/w	S/W DMA2 REQUEST	Software DMA Request on Channel 2 When this bit is set to 1, a software DMA request is initiated on Channel 2. Whether this request is actually recognized depends on the state of AND/OR DMA2. A software DMA request on Channel 2 can be withdrawn by writing a 0 to this bit. This bit defaults to 0 on a channel reset.
0r/w	S/W DMA1 REQUEST	Software DMA Request on Channel 1 When this bit is set to 1, a software DMA request is initiated on Channel 1. Whether this request is actually recognized depends on the state of AND/OR DMA1. A software DMA request on Channel 1 can be withdrawn by writing a 0 to this bit. This bit defaults to 0 on a channel reset.

POS Registers

The POS registers are 8-bit-wide registers that configure the MC-MXI for use on the Micro Channel bus. The local CPU can access these registers by asserting the Micro Channel *CDSETUP* signal on the Micro Channel for the slot containing the MC-MXI. Once *CDSETUP* is asserted, the POS registers are located at addresses 0 through 5. While the Micro Channel specification defines the structure of some of the registers, most of the register structure is adapter dependent. The configuration of the POS registers is contained in an Adapter Description File, which the local CPU accesses during startup. This Adapter Description File is modified when a new Micro Channel adapter is installed. As a result, once you have installed the MC-MXI in a system, you should never have to reconfigure the POS registers.

POS 0/1 MC-MXI Micro Channel ID Register

Function: This register stores the Micro Channel Identification of the MC-MXI. This number is a 16-bit value occupying POS registers 0 and 1. The register is hard wired to generate a value of 0DFFh. This value is the unique ID value that IBM assigned to the MC-MXI.

Address: Micro Channel Read: XXX0h (Low byte)
Micro Channel Read: XXX1h (High byte)

Type: Read Only

Size: 8-bit or 16-bit accessible

Bit Map:

Register 0 (Low byte):

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1

Register 1 (High byte)

7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1

POS 2 MC Setup/DMA Channel 1 Register

Function: This register controls several options on the MC-MXI and sets the arbitration level of the first of the two MC-MXI DMA channels.

Address: Micro Channel Read: XXX2h
Micro Channel Write: XXX2h

Type: Read/Write

Size: 8-bit accessible

Bit Map:

7	6	5	4	3	2	1	0
LBARB	POS1026	0	ARB[3]	ARB[2]	ARB[1]	ARB[0]	CARDEN

Bit	Mnemonic	Description
7r/w	LBARB	Local Bus Arbiter Enable Bit When this bit is set to 1, the Micro Channel bus arbiter of the MC-MXI is enabled. This bit should normally be set to 1. If this bit is enabled, the memory and I/O wait state generation must be set to a minimum of two. Refer to the description of the WAITSTATES register for more details on wait states. LBARB is set to 0 on a channel reset.
6r/w	POS1026	Special Mode Enable bit This bit enables a special mode, which the MC-MXI does not support, on the SMC94C18. You must *always set this bit to 0*.
5	0	RESERVED This bit is reserved for future use. Set this bit to 0.
4–1r/w	ARB1[3–0]	DMA Channel 1 Arbitration Level Bits The MC-MXI supports two DMA channels, each of which has independent arbitration levels on the Micro Channel. These bits set the arbitration level of the first DMA channel, which controls data transfers from the MXIbus to the Micro Channel. The arbitration level ranges from 0 (highest priority) to 15 (lowest priority).

Bit	Mnemonic	Description
0r/w	CARDEN	Card Enable Bit You should set CARDEN to 1 after all the POS registers are properly configured. Setting CARDEN to 1 enables the MC-MXI interface to the Micro Channel. Clearing this bit disables the MC-MXI interface to the Micro Channel, except for POS register configuration. CARDEN is set to 0 on channel reset. CARDEN is required by the Micro Channel specification.

See Also: WAITSTATES Register.

POS 3 Memory/IO Base Address Register

Function: This register sets the MC-MXI base addresses in the I/O and memory address spaces.

Address: Micro Channel Read: XXX3h
Micro Channel Write: XXX3h

Type: Read/Write

Size: 8–bit accessible

Bit Map:

7	6	5	4	3	2	1	0
MEM[3]	MEM[2]	MEM[1]	MEM[0]	I/O[3]	I/O[2]	I/O[1]	I/O[0]

Bit **Mnemonic** **Description**

7–4r/w MEM[3–0] Memory Base Address Bits

MEM[3–0] sets the base address of the MC-MXI in the Micro Channel memory address space. These bits dictate the base memory addresses of the three MC-MXI windows.

Setting MEM[3] to 1 enables a single window on the MC-MXI. This window will be located in the first 1024 KB of system memory. The base address of the window is determined by the MEM[2–0] bits as shown below:

19	18	17	16	15 0
1	MEM[2]	MEM[1]	MEM[0]	All 0s

Only address bits A0–A19 are recognized in this mode. Use this mode when running DOS.

Clearing MEM[3] enables all three windows. The base addresses of these windows are determined by the value of the MEM[2] and MEM[1] bits. The following table shows the available base addresses:

MEM[2]/MEM[1]	0/1	1/0	1/1
Window 1	79FF0000	B9FF0000	F9FF0000
Window 2	7A000000	BA000000	FA000000
Window 3	7B000000	BB000000	FB000000

Bit	Mnemonic	Description
		Window 1 is always 64 KB long while Windows 2 and 3 are 16 MB long.
3–0	r/w I/O[3–0]	Input/Output Base Address Bits I/O[3–0] dictates the base address of the MC-MXI registers (except the POS registers) in the Micro Channel I/O space. The base I/O address of the MC-MXI is determined by left-shifting the value of I/O by 12 bits and ORing this shifted value with 0000h. In effect, the value of I/O denotes the four most significant bits of a 16-bit address. As a result, the registers of the MC-MXI can be located at 16 evenly spaced 4-KB intervals in the Micro Channel I/O address space.

POS 4 MC Interrupt Map Register

Function: This register maps different groups of MC-MXI interrupts to specific Micro Channel interrupt levels.

Address: Micro Channel Read: XXX4h
Micro Channel Write: XXX4h

Type: Read/Write

Size: 8-bit accessible

Bit Map:

7	6	5	4	3	2	1	0
COMM[1]	COMM[0]	MXIRQ[1]	MXIRQ[0]	MISCINT[1]	MISCINT[0]	MMERR[1]	MMERR[0]

Bit **Mnemonic** **Description**

7–6r/w COMM[1–0] Communications Interrupt Map Bits

The COMM[1–0] bits map the interrupt signal generated by either message passing or Word Serial Protocols. The operations that trigger the COMM interrupt are enabled by the Communication Control register. The value of COMM[1–0] determines which Micro Channel interrupt signal is mapped to the COMM interrupt. The following table lists these values:

COMM[1]	COMM[0]	Micro Channel Interrupt
0	0	IRQ*10
0	1	IRQ*11
1	0	IRQ*15
1	1	IRQ*5

5–4r/w MXIRQ[1–0] MXIbus Interrupt Map Bits

These bits map the MXIbus *IRQ** signal to a Micro Channel Interrupt. The mapping function is the same for the COMM interrupt.

Bit	Mnemonic	Description
3–2r/w	MISCINT[1–0]	General Interrupts Bits This interrupt signal is driven by three different events: terminal count (TCINT), MXIbus ownership (OWNMBINT), and requests for Micro Channel accesses (PCREQINT). Setting the appropriate bits in the MC-MXI–CTRL register enables this type of interrupt. It is possible to set more than one of these general interrupts at a time, but these interrupts are still connected to only one Micro Channel interrupt signal. The mapping function is the same for the COMM interrupt.
1–0r/w	MMERR[1–0]	Error Condition Bits This interrupt is triggered by error conditions on the MC-MXI. Examples of such errors are parity error on the MXIbus and receipt of BERR on the MXIbus from a remote master or slave device. The mapping function is the same for the COMM interrupt. **Note:** ***You can set up COMM, MXIRQ, MISC, and MMERR interrupts to share the same Micro Channel IRQ* signal by writing the same 2-bit codes to the different interrupt classes.***

See Also: Communication Control Register, MC-MXI–CTRL Register.

POS 5 MC Arbitration/DMA Channel 2 Register

Function: This register controls several options on the MC-MXI and sets the arbitration level of the second of the two MC-MXI DMA channels.

Address: Micro Channel Read: XXX5h
Micro Channel Write: XXX5h

Type: Read/Write

Size: 8-bit accessible

Bit Map:

7	6	5	4	3	2	1	0
CHCK*	CHCKSTAT	0	FAIR	ARB[3]	ARB[2]	ARB[1]	ARB[0]

Bit	Mnemonic	Description
7r/w	CHCK*	Channel Check Bit This bit is active low and is set by the MC-MXI when an error occurs that affects the Micro Channel. When this bit is set low, the MC-MXI generates a channel check interrupt on the Micro Channel. This bit is set high on reset, and it should be set high for normal operations.
6r/w	CHCKSTAT	Channel Check Status Bit The SMC94C8 provides this bit to indicate the presence of an optional channel check status word. Because the MC-MXI has no such status word, you must *always set this bit to 0.*
5	0	RESERVED
4r/w	FAIR	Fairness Bit A 0 on this bit directs the DMA controllers to participate in the next arbitration cycle on the Micro Channel. A 1, however, instructs the DMA controllers to wait until all other devices on the Micro Channel have finished accessing the bus. This fairness feature applies only when other Micro Channel devices request access by asserting the *PREEMPT** signal on the Micro Channel.
3–1r/w	ARB2[3–0]	DMA Channel 2 Arbitration Level Bits The MC-MXI supports two DMA channels, each of which has independent arbitration levels on the Micro Channel. These bits set the arbitration level of the second DMA channel, which controls data transfers from the Micro Channel to the MXIbus. The arbitration level ranges from 0 (highest priority) to 15 (lowest priority).

Chapter 5
Programming the MC-MXI

This chapter discusses important initialization and normal-operation programming aspects of the MC-MXI. The MC-MXI operating characteristics depend on interaction with software. To use the MC-MXI, your application software must initialize the MC-MXI board. When the MC-MXI is initialized, you need to program it to transmit data to and from the MXIbus.

This chapter refers to many of the registers described in Chapter 4, *Register Descriptions*. You should refer to these register descriptions as necessary. Some registers contain bit fields that you must set to specific values. These required bit fields are listed under the appropriate register descriptions in Chapter 4.

This chapter uses pseudo-code to present programming examples. The pseudo-code instructions encompass basic read and write operations as shown in Table 5-1.

Table 5-1. Pseudo-Code Instructions

Pseudo-Code Instruction	Operation
`outb` *`address, data`*	Writes a single byte of data to the specified address.
`read` *`address`*	Reads two bytes of data starting at the designated address.
`write` *`address, value`*	Writes a two-byte value to the designated address, beginning at the designated address.
`inb` *`variable, address`*	Reads a single byte from the specified address and stores the value in the designated variable.

Initializing the MC-MXI

You must initialize the MC-MXI before you can use it as a MXIbus device. When power is applied to the MC-MXI, it defaults to an inactive state. During this inactive state, all MC-MXI logic is disabled except for access to the MC-MXI POS registers. As a result, the MC-MXI is accessible from the Micro Channel only through the POS registers, and it is completely inaccessible from the MXIbus. To properly initialize the MC-MXI, you must perform the following steps:

1. Set up the MC-MXI POS registers. This step enables the DMA channels, maps interrupts to the Micro Channel, and specifies the MC-MXI location in the Micro Channel I/O and memory address spaces.

2. Specify memory and I/O wait states, and enable the MC-MXI System Controller base clock.
3. Specify the cycle period of the MC-MXI System Controller base clock.
4. Enable and initialize the MC-MXI timers.
5. Enable the MXIbus System Controller (optional).

Step 1. Set Up the MC-MXI POS Registers

During startup, the local CPU loads the MC-MXI POS registers with information contained in a data file. The contents and location of this data file is computer dependent. For example, on an IBM PS/2, the POS register information is located in an Adapter Description File, while the IBM RS6000 stores this information in a device driver. The MC-MXI supports POS registers 0 through 5. Refer to Chapter 4 for detailed descriptions of the POS registers.

To access the POS registers, your host CPU must assert the *CDSETUP* signal for the Micro Channel slot containing the MC-MXI and drive a base address onto the Micro Channel. The original Micro Channel specification (circa 1987) located the POS registers at base address 0100 (hex). MC-MXI boards from Rev A through Rev B.3 locate their POS registers at this base address. Later revisions to the Micro Channel specification changed this requirement to place the base address of the POS registers at XXX0. Versions of the MC-MXI beyond B.3 follow this arrangement. If you have an older MC-MXI, you can program it to recognize the XXX0 addressing convention by writing a 0 to the CARDEN bit in POS register 2 before you read and write to the other POS registers. Most IBM PS/2 and RS6000 computer systems access the POS registers by using a base address of 100. Some models of the RS6000 use both 0100 and 0000 base addresses to access POS registers during different operating modes.

POS registers 0 and 1 are read only and contain a unique 16-bit identification number. POS register 0 contains the low-order byte of this identification number, and POS register 1 contains the high-order byte. The MC-MXI identification number is 0DFF (hex).

POS registers 2 and 5 control several different features, but their primary purpose is to set the arbitration levels of the two MC-MXI DMA channels. POS register 2 controls the arbitration level of the first DMA channel, which transfers data from the MXIbus to the Micro Channel. POS register 5 determines the arbitration level of the second DMA channel, which transfers data from the Micro Channel to the MXIbus. The arbitration level can range from 0 (highest priority) to 15 (lowest priority).

POS register 3 determines the base address of the MC-MXI registers in Micro Channel I/O space and the base addresses of the MC-MXI memory windows in Micro Channel memory space. The description of the Memory/IO Base Address register (POS register 3) in Chapter 4 explains how to configure the MC-MXI base addresses. However, you should consider the following points:

- If the host computer is an 80x86 microprocessor running DOS, the MC-MXI supports one 64-KB window within the first 1 MB of system memory. Therefore, always write a 1 to bit 7 of POS register 3 when running DOS.

- If the host computer is an 80x86 microprocessor operating in protected mode or if the host computer is an AIX workstation, the MC-MXI supports three memory windows. The first window is 64 KB long while the other two windows are 16 MB long. As a result, write a 0 to bit 7 of POS register 3.

POS register 4 maps interrupts generated on the MC-MXI to different interrupt levels on the Micro Channel. The MC-MXI generates four different types of interrupts:

- COMM interrupts are generated by communication with other MXIbus devices.
- MXIRQ interrupts are generated when the MXIbus *IRQ** signal is asserted.
- MISCINT interrupts are initiated by different events inside the MC-MXI.
- MMERR interrupts are generated by error conditions in the MC-MXI.

The MC-MXI interrupts can map to IRQ*5, IRQ*10, IRQ*11, or IRQ*15 on the Micro Channel by using a 2-bit mapping code for each type of interrupt. The description of POS register 4 in Chapter 4 discusses how to perform this mapping. You can map different types of MC-MXI interrupts to the same *IRQ*x* signal on the Micro Channel. To do this, simply use the same 2-bit mapping code for each interrupt type.

Note: ***When you are configuring the MC-MXI POS registers, clear the CARDEN bit in POS register 2 prior to configuring the POS registers. Set the CARDEN bit after configuring the POS registers. This precaution will prevent the MC-MXI from operating erratically when sudden changes occur in its POS registers.***

Step 2. Specify MC-MXI Wait States

The MC-MXI incorporates a 94C18 Micro Channel interface chip. You must set this chip to a special mode, and in order to set this chip to a special mode, you must specify memory and I/O wait states. These wait states give the MC-MXI bus arbitration circuitry time to resolve simultaneous requests to and from the MC-MXI. The WAITSTATES register determines the number of I/O and memory wait states. You must set the register for two wait states for I/O and two wait states for memory. The following sample code shows how to specify the number of wait states:

```
outb  WAITSTATES, 0x6d; enable timer function and program for 2 wait
      states
```

In general, once you write 0x6d to the WAITSTATES register, you do not need to access this register again until a channel reset occurs. This example also enables the MXIbus System Controller base clock (SYSCLK). The following section describes how to set the period of SYSCLK.

Step 3. Specify the Cycle Period of the System Controller Base Clock

The System Controller base clock (SYSCLK) is a constant clock that serves as the base time increment for measuring System Controller timeout periods. The System Controller timeout gives a MXIbus System Controller the means to terminate a MXIbus data transfer if a MXIbus device fails to respond to a data transfer. While you have the option of enabling the MC-MXI System Controller capability, you must *always* enable and initialize the System Controller's base clock.

You enabled SYSCLK when you specified the number of wait states by following the steps in the preceding section, but you must still specify the SYSCLK cycle period. Set the cycle period of SYSCLK to 800 ns by writing EF (hex) to the SYSCLK register.

If for some reason you need to change this value, use the following formula to determine the clock frequency:

$$\frac{20\ \text{MHz}}{16 - T(7\text{–}4))} * 8$$

where T(7–4) represents the most significant bits of the SCCLK register.

Note: ***While you can change the System Controller timeout period by altering the period of SYSCLK, you should not do this. Instead, alter the System Controller timeout period by using the PITSC timer as described in the following section.***

Step 4. Initialize the Timers

Three timers monitor different MC-MXI interface functions. These timers terminate data transfers if the elapsed time from the beginning of the transfers exceeds a specified value. This time limit is called a *timeout*. Each timer generates a different timeout for a different type of data transfer.

- A *local* timeout terminates MXIbus slave-mode data transfers from the MXIbus to the Micro Channel. The PITLOCAL timer controls this timeout.

- A *ready* timeout terminates MXIbus master-mode data transfers from the Micro Channel to the MXIbus. The PITREADY timer controls this timeout.

- A *System Controller* timeout terminates data transfers on the MXIbus that are initiated by any MXIbus master. The PITSC timer controls this timeout. Notice that the System Controller timeout is operational only if the MC-MXI is enabled as a MXIbus System Controller.

You must enable and program each timer before you use its associated function. In general, it is a good idea to program all timers during MC-MXI initialization and prior to any MXIbus activity. The PITCNTR register enables the timers. In all cases, write the following three bytes to the PITCNTR registers:

```
outb PITCNTR, 34(hex)
outb PITCNTR, 74(hex)
outb PITCNTR, b4(hex)
```

The description of the PITCNTR register in Chapter 4 explains the purpose of these bytes. You should write these bytes in the order shown. In general, these are the only bytes you will ever write to this register.

You must write a count value to the PITSC, PITLOCAL, and PITREADY registers to program them with the appropriate time periods. Each timer contains a 16-bit counter that decrements at a rate equal to the base clock frequency of the timer. When a data transfer begins, the appropriate timer begins decrementing. If the count reaches 0 and the data transfer is not complete, the timer generates a timeout signal. The counter reloads with its original programmed value and the process repeats. As mentioned in Chapter 4, the PITSC, PITLOCAL, and PITREADY registers are 8 bits wide, yet they service 16-bit counters. To load the counters, write the least significant byte to the appropriate register first, then write the most significant byte to the appropriate register. The timers automatically place the bytes into the correct locations in their respective counters.

Table 5-2 contains information that you will need to program the PITSC, PITLOCAL, and PITREADY timers. To determine the time period of each timer, multiply the base clock period by the number of counts programmed into the timer's counter. Table 5-2 lists suggested values and their corresponding time periods. In general, you should use the recommended values to program the timers.

Table 5-2. Recommended Count Values for MC-MXI Timers

Timer	Base Clock Period (ns)	Suggested Count (hex)	Suggested Time Period
PITSC	800 *	0x4E2	1.0 ms
PITLOCAL	100	0x4C	7.6 µs
PITREADY	100	0x23	3.5 µs
* Assumes the System Controller base clock frequency is 1.25 MHz			

The MC-MXI gives you the capability to read the contents of the PITLOCAL, PITSC, and PITREADY timers. Refer to the descriptions of the timer registers in Chapter 4 for a discussion of this procedure.

Step 5. Enable the MXIbus System Controller

Assuming the PITSC timer and the System Controller base clock are properly configured, you can now enable the MC-MXI MXIbus System Controller capabilities. To enable these capabilities, set the MBSC bit in the MC-MXI–CTRL register. Remember that if you want the MC-MXI to operate as a System Controller, it must be the first device on the MXIbus.

Operating the MC-MXI

This section describes how to control the MC-MXI in order to transfer data between devices on the Micro Channel and devices on the MXIbus. The MC-MXI transfers data between the two bus architectures by acting as a master or slave on each bus, as appropriate. Further, the MC-MXI transmits interrupts between the Micro Channel and MXIbus. The remainder of this chapter discusses the following MC-MXI operating issues:

- MXIbus master-mode operation
- MXIbus slave-mode operation
- Controlling interrupts between the MXIbus and Micro Channel
- Byte swapping

MXIbus Master-Mode Operation

MXIbus masters initiate transactions on the MXIbus. When the MC-MXI functions as a MXIbus master, it simultaneously functions as a Micro Channel slave. Before attempting to use the MC-MXI as a MXIbus master, you must enable MXIbus master-mode operation by setting the MMEN bit in the MC-MXI–CTRL register.

The MC-MXI has three separate windows to access MXIbus memory from the Micro Channel. One window (window 1) is 64 KB in size. The other two windows (windows 2 and 3) are 16 MB in size. Each of the three windows has its own address modifier register, which is driven on the MXIbus address modifier lines when a MXIbus access is performed through that window. As a result, you can set up three areas in Micro Channel memory space, and each area can access a different MXIbus address space (A16, A24, and A32). Having three windows is beneficial because you do not have to alter the address modifier register if you are accessing different address spaces.

Although the MC-MXI supports three unique windows and address modifier registers, you do not have to set them to the different MXIbus address spaces. There are no restrictions on how to interpret or use the window spaces. For example, the address modifier registers of both 16-MB windows could access A32 space. This arrangement gives you direct access to 32 MB of A32 through these windows.

However, if you want direct access to all three MXIbus address spaces, configure window 1 to A16 space, window 2 to A24 space, and window 3 to A32 space. This arrangement gives you direct access to all of A16 space, all of A24 space, and 16 MB of A32 space.

The following code shows how to set up the address modifier registers so that window 1 is set to A16 space (nonprivileged), window 2 is set to A24 space (nonprivileged data), and window 3 is set to A32 space (nonprivileged data):

```
outb  WIN1AM-SMOFF, 0x01100; set window 1 to A16 address nonprivileged.
outb  WIN3AM-WIN2AM, 0x0119; set window 3 to A32 nonprivileged and window
                             2 to A24 nonprivileged
```

You can rewrite the address modifier codes at any time; as a result, you can dynamically change window spaces and access privileges.

Operating the MC-MXI under a DOS Environment

You cannot use windows 2 and 3 while running under DOS. The DOS operating system places the processor in real mode and restricts direct access to the lowest one megabyte of system memory. When running DOS, you must use window 1, which maps into the first megabyte of Micro Channel memory, to access all MXIbus resources. Consequently, you must write to the WIN1AM-SMOFF register whenever you change the MXIbus address space (A16, A24, or A32) and/or access privilege (supervisory, nonprivileged, or block).

Accessing MXIbus Memory in Master Mode

When operating as a MXIbus master, the MC-MXI automatically translates Micro Channel cycles into MXIbus cycles. The translation operation is transparent and requires no additional software. However, you might benefit in some situations by adding code to incorporate further MXIbus functionality, such as the following:

- Memory paging: This capability is useful when you want a memory window that you can move or *page* around in a larger system memory space. By paging, you can access a large memory space while using a smaller portion of the local Micro Channel address space.
- Deadlock resolution: Deadlocks can occur when other MXIbus masters access resources on the Micro Channel bus asynchronously while Micro Channel bus masters simultaneously access resources on the MXIbus. If a deadlock occurs, the MC-MXI prematurely terminates the Micro Channel cycle so that the other cycle can complete. Your application software must then retry the Micro Channel cycle access.
- Coupling dissimilar buses together: The timing parameters of a remote bus servicing another MXIbus master device might not adhere to the timing restrictions imposed by the Micro Channel that services the MC-MXI. The PITSC, PITLOCAL, and PITREADY timers prevent timing violations by terminating data cycles that exceed the time limit imposed by the timers. If the MC-MXI terminates a data cycle, your application software should retry the cycle.
- Block-mode data transfers: The MXIbus can rapidly transfer contiguous segments of data by using a block-mode transfer. The advantage of this type of data transfer is that the MC-MXI does not have to initiate a new MXIbus data transfer cycle for each piece of data.

The following sections discuss these situations and the software requirements in more detail.

Memory Paging

The MC-MXI has an onboard Master-Mode PAGE register to move master-mode windows around in large memory spaces. Paging is required when you want to access more than 64 KB of A24 or A32 space while running DOS or when you want to access more than 16 MB of A32 space by using window 2 or 3. The Master-Mode PAGE register supplies the additional address signals required for any MXIbus accesses that the Micro Channel does not provide.

The Master-Mode PAGE register is 16 bits wide, but the number of bits actually applied to an address depends on the memory window you use and the MXIbus memory space you access. Table 5-3 illustrates how the Master-Mode PAGE register and the address provided from the Micro Channel combine to form a MXIbus address.

Table 5-3. MXIbus Address Formation Using the Master-Mode PAGE Register

Type of Access		**Composition of MXIbus Address**			
Window	**Address Space**	**A31 A24**	**A23 A16**	**A15 A8**	**A7 A0**
1	A16	Not used		MC Addr[15–0] *	
1	A24	Not used	PAGE[7–0]	MC Addr[15–0]	
1	A32	PAGE[15–0]		MC Addr[15–0]	
2	A16	Not used		MC Addr[15–0]	
2	A24	Not used	MC Addr[23–0]		
2	A32	PAGE[15–0]	MC Addr[23–0]		
3	A16	Not used		MC Addr[15–0]	
3	A24	Not used	MC Addr[23–0]		
3	A32	PAGE[15–0]	MC Addr[23–0]		

* *MC Addr* is the address provided by the Micro Channel address bus.

The PAGE register is not required when you want to access A16 space through window 1 or A24 space through windows 2 and 3. Window 1, which is 64 KB in size, is the same size as A16 space. Windows 2 and 3, which are 16 MB in size, are the same size as A24 space. As a result, you do not have to use the PAGE register when you use these combinations of windows and address spaces.

The following program example, which illustrates how the PAGE register operates, assumes you are working under DOS, in which only window 1 is available. The code demonstrates how you set up the address modifier and page registers to perform the following tasks:

1. Read MXIbus location 4321h in A16 space.
2. Change to A24 space and write MXIbus location 123456h.
3. Change to A32 space and read location 87654321h.

The following code assumes the 64-KB window is located at segment C000h:

```
outb  WIN1AM-SMOFF, 0x11  ; set to A16 address modifier code
read  0xC4321

outb  WIN1AM-SMOFF, 0x19  ; set to A24 address modifier code
outb  PAGE, 0x12          ; set up page register
write 0xC3456, value

outb  WIN1AM, 1           ; set to A32 address modifier code
outw  PAGE, 0x8765        ; set up page register
read  0xC4321
```

After you program the PAGE register and address modifier for the window you want to use, you should rewrite them only if you need to access a different address space or if you need to access memory that is outside the segment available in the window space.

Deadlock

Because the MXIbus is a multimaster bus, you can build systems that incorporate multiple processors or DMA controllers (masters). These processors, in turn, can connect to one another's resources over the same global MXIbus network. This system configuration is advantageous because multiple masters can operate in parallel and directly access one another's shared resources without the overhead of software communication protocols. The only time the individual processors must synchronize their activities is when one of the MXIbus masters accesses the resources of another MXIbus master.

When multiple processors operate in parallel without a global arbitration mechanism, two or more processors may simultaneously attempt to access each other's resources, thus deadlocking the system. Keep in mind that deadlocks are possible only if multiple processors are allowed to access *each other's* resources *asynchronously*. Simultaneous asynchronous transfers can be avoided by a software message-passing protocol that precludes deadlock by coordinating activity among the MXIbus masters. Without such protocols, MXIbus devices must support logic to detect deadlocks.

The MC-MXI has onboard logic that can sense a deadlock condition and automatically back off (terminate) a Micro Channel master's (that is, the local CPU's) access to the MC-MXI. The MC-MXI can then signal the Micro Channel master that a deadlock condition has occurred. The MC-MXI signals the Micro Channel master by either setting the DL bit in the MC-MXI–STATUS register or by generating a Micro Channel interrupt. The Micro Channel master can then retry the operation that was terminated. The onboard deadlock detection logic will always terminate a Micro Channel Access to the MC-MXI during deadlock situations.

The following is an example of a routine that checks for a deadlock after a master-mode write transfer and retries the access if a deadlock occurred:

```
again   write     address, value
        inb       status, MC-MXI-STATUS
        if        DL bit is set in status then go to again
```

Optionally, the MC-MXI can generate an interrupt on a deadlock condition so that additional code executes only when deadlocks occur. The deadlock condition interrupt is one of the master mode error (MMERR) interrupts, and it is discussed further in the *Controlling Interrupts between the MXIbus and the Micro Channel* section, which is located later in this chapter.

Timing Incompatibilities

Because the MXIbus lets you couple dissimilar bus structures together, one bus might impose timing restrictions that another bus structure cannot meet. For example, the Micro Channel bus requires that slave devices take no longer than 3.5 µs to respond to a transfer. However, if the Micro Channel is coupled to the VMEbus (via a VME-MXI), which imposes no such access-time restrictions on its slave, the Micro Channel access cycle could potentially timeout before the VME slave has responded to the transfer. If you cannot guarantee that the slave can respond to a Micro Channel access (including all worst case arbitration latencies), you may need to insert code that will retry the access in the case of a ready timeout into your program. This condition is called a *ready timeout* because it refers to the amount of time the MC-MXI can unassert the *READY* signal on the Micro Channel in order to extend the transfer.

As with the deadlock condition, the MC-MXI has onboard logic that can sense a ready timeout condition and automatically back off (terminate) the access of the local CPU. The MC-MXI can then signal the master that a timeout occurred. The MC-MXI signals the master by either setting the TO bit in the MC-MXI–STATUS register or by generating a Micro Channel interrupt. The local CPU can then retry the operation.

The following is an example of a routine that checks for a timeout after a master-mode read transfer and retries the access if a timeout occurs:

```
again   read      value, address
        inb       status, MC-MXI-STATUS
        if        TO bit is set in status then go to again
```

Programming the MC-MXI for Block-Mode Transfers

If a Micro Channel master transfers large blocks of data from or to another MXIbus device and the data is in consecutive memory locations, you can program the MC-MXI to transfer the data by using block-mode cycles on the MXIbus. MXIbus block-mode transfers are faster than normal cycles because addressing information does not have to be sent over the MXIbus with every cycle access.

There are two methods of performing a block-mode transfer. The first method uses the local CPU to perform programmed I/O accesses to the MC-MXI. The second method uses a DMA controller that is resident on the Micro Channel to transfer data to the MC-MXI. The two methods require different sequences of steps.

The following steps describe how you perform MXIbus block-mode transfers by using programmed I/O:

1. Select the window to use for the transfer.
2. Set the address modifier bits for this window for a block-mode transfer (see Table 1-3, *Address Modifier Codes*, in Chapter 1, *Introduction to MXIbus*).
3. Set the MBBLOCKEN bit in the MC-MXI–CTRL register to 1.
4. Read/write the first data element in the block.
5. Read/write the remainder of the data elements (except for the last element).
6. Clear the MBBLOCKEN bit.
7. Read/write the last data element.

The following steps describe how you perform block-mode data transfers by using a DMA controller:

1. Select the window to use for the transfer.
2. Set the address modifier bits for this window for a block-mode transfer (see Table 1-3).
3. Set up the DMA controller on the Micro Channel that will control the data transfer. Configure the DMA controller to transfer all the data except for the first element.
4. Set the MBBLOCKEN bit in the MC-MXI–CTRL register to 1.
5. Set the MDRQEN bit in the MC-MXI–CTRL register. Setting this bit enables the MC-MXI DMA transfer capability.
6. Set the TCIE bit in the MC-MXI–CTRL register. Setting this bit causes the MC-MXI to terminate the block-mode operation when the DMA controller is finished transferring data.
7. If you want the MC-MXI to generate an interrupt after the block transfer is finished, set the TCIE bit in the MC-MXI–CTRL register.
8. Read/write the first element of data. This *primes* the MC-MXI. The MC-MXI can then generate DMA requests on the Micro Channel.
9. If the TCIE bit is set, the MC-MXI generates an interrupt when the block transfer completes. Otherwise, monitor the DMA controller to determine when the transfer is finished.
10. Clear the MDRQEN bit.
11. Clear the MBBLOCKEN bit.

In general, the programmed I/O method tends to transfer data faster than the DMA method because modern CPUs can transfer data faster than DMA controllers. However, the penalty for this high speed is that the CPU must devote all its resources to the data transfer. You should consider the benefits and drawbacks of each method when you determine which approach will best suit your needs.

MXIbus Slave-Mode Operation

When the MC-MXI operates as a MXIbus slave, it converts memory accesses from the MXIbus into memory accesses into the Micro Channel memory space. The MC-MXI has a memory window from MXIbus A24 or A32 space to the Micro Channel and another window from MXIbus A16 space into the MC-MXI registers. Most of these operations are transparent to the local CPU, but you must understand how to determine which portion of MXIbus memory maps into Micro Channel memory. The process of setting up the MC-MXI as a MXIbus slave device involves the following five steps:

1. Set the MC-MXI logical address to specify the location of the MC-MXI registers in A16 space. Setting the MC-MXI logical address determines the location of the MC-MXI register set in A16 space. You specify the logical address by writing to the MXI Logical Address register. Only the MC-MXI registers map to A16 space.

2. Configure the MC-MXI slave-mode window to reside in either MXIbus A24 or A32 space. The MC-MXI supports one slave-mode window, which maps to either A24 or A32 space. The ADSP bits in the MXI ID register determine the location where the MC-MXI will map in MXIbus address space. Writing 00b to the ADSP bits places the slave window in A24 space, and writing 01b places the slave window in A32 space. You can also disable the slave window by writing 11b to the ADSP bits.

3. Specify the amount of MXIbus memory the MC-MXI will require. The RQMEM bits in the MXI Device Type register control the size of the slave window. The RQMEM bits operate differently, depending on whether the slave window maps to A24 or A32 space. When the slave window maps to A24 space, RQMEM determines the size of the window based on the following formula:

   $$\text{Size} = 2^{(23 - m)}$$

   where m is the decimal equivalent of the value written to RQMEM.

   If the slave window maps to A32 space, use the following formula to determine the window size:

   $$\text{Size} = 2^{(31 - m)}$$

   As the formula shows, the size of the slave windows varies inversely with the value of RQMEM. RQMEM is a 4-bit value that ranges from 0 to 15 (decimal). In all cases, the maximum size of the slave window is half the total size of the address space. As a result, the maximum window size in A24 space is 8 MB, and the maximum size in A32 space is 2 GB.

4. Specify the base address of the MC-MXI in the MXIbus address space. The MXI Offset register, in conjunction with the value of RQMEM, specifies the base address. The MXI Offset register contains a 16-bit value that comprises the most significant bits of the base address. However, the value of RQMEM determines how many bits in the MXI Offset register the MC-MXI uses to create a base address. The (RQMEM[] + 1) most significant bits of the MXI Offset register form the (RQMEM[] + 1) most significant bits of the base address. The remaining bits of the MXI Offset register are meaningless.

   The following example shows how the MXI Offset register and the value of RQMEM form a base address in A24 space. In this example, RQMEM = 3, which means that the size of the slave window is 1 MB. The four most significant bits (that is, 3 + 1) of the MXI Offset register serve as the four most significant bits of the base address.

23 20	19 0
Offset[15–12]	All 0s

   While determining the base address of the slave-mode window is somewhat intricate, the slave-mode window does follow a logical path. In the preceding example, RQMEM sets the size of the slave window to 1 MB, which results in 16 potential locations for the slave window in A24 space. Because four bits in the MXI Offset register apply toward the base address, the slave window can be placed at 16 different locations in A24 space.

5. Specify the base address of the MC-MXI slave window in Micro Channel memory space. The method for setting the base Micro Channel address that corresponds to the base MXIbus address of the slave-mode window involves the use of the WIN1AM-SMOFF register. If the slave window maps to MXIbus A32 space, the SM32OFF bits in the WIN1AM-SMOFF register comprise the four most significant bits of the Micro Channel base address (A28 through A31). Similarly, the SMOFF24 bits in the WIN1AM-SMOFF register comprise the most significant bits of a 24-bit base Micro Channel address (A20 through A23).

   As a result, when the slave window maps to A32 space, the base Micro Channel address can be placed at 16 different locations in Micro Channel memory on 256-MB boundaries. If the slave window maps to A24 space, the base Micro Channel address can be placed at 16 different locations on 1-MB boundaries within the first 16 MB of Micro Channel memory.

A problem with accessing the Micro Channel memory space is that the MXIbus is mapping two separate memory spaces (A24 and A32 space) into the one memory space on the Micro Channel. The MC-MXI solves this problem by mapping MXIbus A24 space into the first 16 MB of Micro Channel memory space and mapping A32 space directly to the Micro Channel memory space. However, this arrangement prevents the MC-MXI from recognizing A32 accesses that are mapped to the first 16 MB of Micro Channel memory.

Controlling Interrupts between the MXIbus and the Micro Channel

The MC-MXI has an interrupt scheme in which any of 12 different conditions generates an interrupt. The MC-MXI groups these 12 interrupt conditions into four separate interrupt signals. These four signals can map to four different Micro Channel interrupts. The four interrupt signals group the interrupt conditions according to function.

- *Communication* (COMM) interrupts result from message passing between the MC-MXI and other MXIbus devices. These interrupts are part of the MC-MXI's role as a VXI Message-Based device. The COMM interrupt is the logical OR of four different conditions:
  - The *Non-Empty Signal register* condition exists whenever the Signal register FIFO is not empty.
  - The *Word Serial Read Request* condition exists when the Data (Out) registers do not contain data.
  - The *Word Serial Write Request* condition exists when the Data (In) registers contain data.
  - The *Soft Reset* condition exists when the RESET bit in the MXI Control register is set to one.
- *Miscellaneous* (MISC) interrupts result from three different conditions that the MC-MXI generates internally:
  - The *PC Request* (PCREQ) condition exists when a MXIbus device requests access to the Micro Channel. This condition is useful because it can alert the local CPU that an external device is attempting to gain access to the Micro Channel.
  - The *Own MXIbus* (OWNMB) condition exists when the MC-MXI wins control of the MXIbus as a MXI master.
  - The *Terminal Count* (TC) condition exists when the MC-MXI finishes a MXIbus block-mode operation. This condition serves as a means to alert the local CPU when the MC-MXI completes a DMA transfer.
- *Master-Mode Error* (MMERR) interrupts result from four different error conditions that the MC-MXI generates:
  - The *Parity Error* (PERR) condition exists when the MC-MXI detects a parity error on the MXIbus *AD[31–0]** signals.
  - The *MXIbus Error* (BERR) condition exists when the *BERR** signal on the MXIbus is asserted.
  - The *Timeout* (TO) condition exists when one of the three MC-MXI timers generates a timeout signal in response to a data transfer that did not complete within a specified time.
  - The *Deadlock* (DL) condition exists when another MXIbus device attempts to access the Micro Channel at the same time that the MC-MXI attempts to access the MXIbus.
- *MXIbus* (MXIRQ) interrupts trigger when the MXIbus *IRQ** signal is asserted.

Creating an interrupt on the Micro Channel consists of two phases. The first phase consists of configuring the MC-MXI to recognize one of the four different interrupt signals (*COMM*, *MISC*, *MMERR*, or *MXIRQ*). The second phase consists of configuring the MC-MXI to generate an interrupt on the Micro Channel. The following sections explain these two phases in more detail.

Recognizing COMM Interrupts

The MC-MXI recognizes COMM interrupts by manipulating the Communications Control register and monitoring the Communications Status register. The Communications Control register contains four enable bits. Each bit enables one of the four COMM interrupt conditions. If a COMM interrupt condition occurs and the corresponding enable bit in the Communications Control register is set to 1, a *COMM* interrupt signal generates. The Communications Status register contains the state of the four enable bits so that you can confirm the state of the COMM interrupt enable bits. The conditions that generate a COMM interrupt are not mutually exclusive. If the COMM interrupt is valid, more than one interrupt condition may exist. The enable bits *do not* reset after an interrupt condition occurs.

Figure 5-1 depicts the structure of the interrupt recognition logic for COMM interrupts.

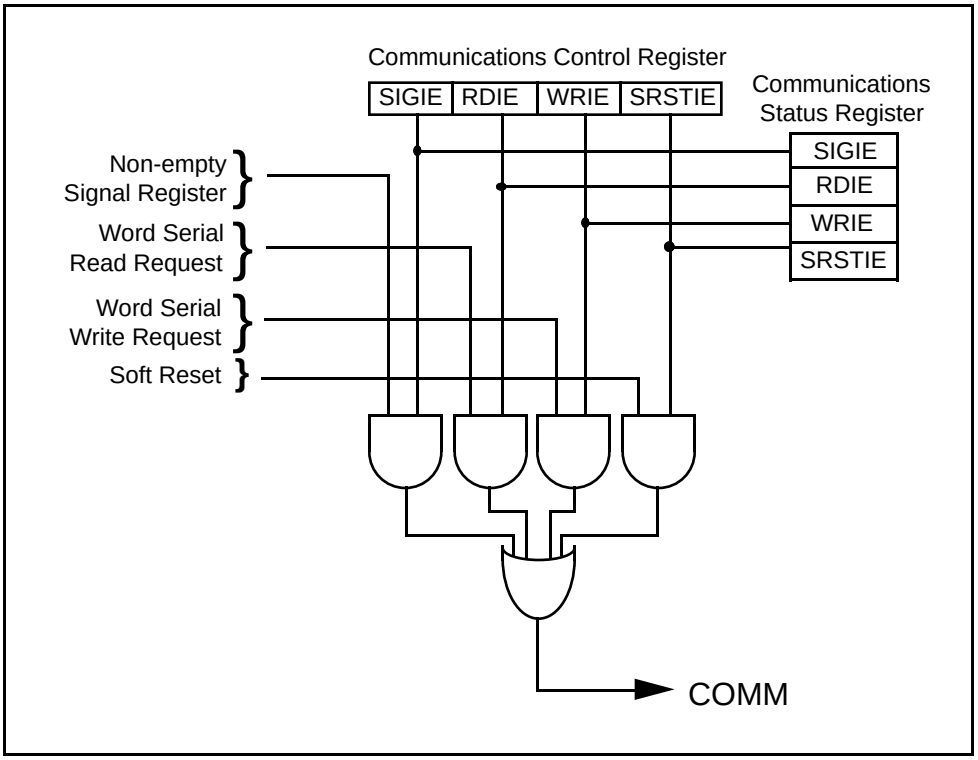


Figure 5-1. Logic to Recognize COMM Interrupts

Recognizing MISC Interrupts

Recognizing MISC interrupts is similar to recognizing COMM interrupts. The MC-MXI–CTRL register contains three enable bits: one for each type of MISC interrupt condition. If a MISC interrupt condition occurs and the corresponding enable bit in the MC-MXI–CTRL register is set to 1, a *MISC* interrupt signal generates. The MC-MXI–STATUS register contains three bits that reflect the state of the three MISC interrupt conditions. As Figure 5-2 indicates, some of the conditions that trigger the MISC interrupt also set bits in the MC-MXI–STATUS register. Keep in mind that these status bits are valid as long as the interrupt condition exists.

Figure 5-2 depicts the structure of the interrupt recognition logic for MISC, MMERR, and MXIRQ interrupts.

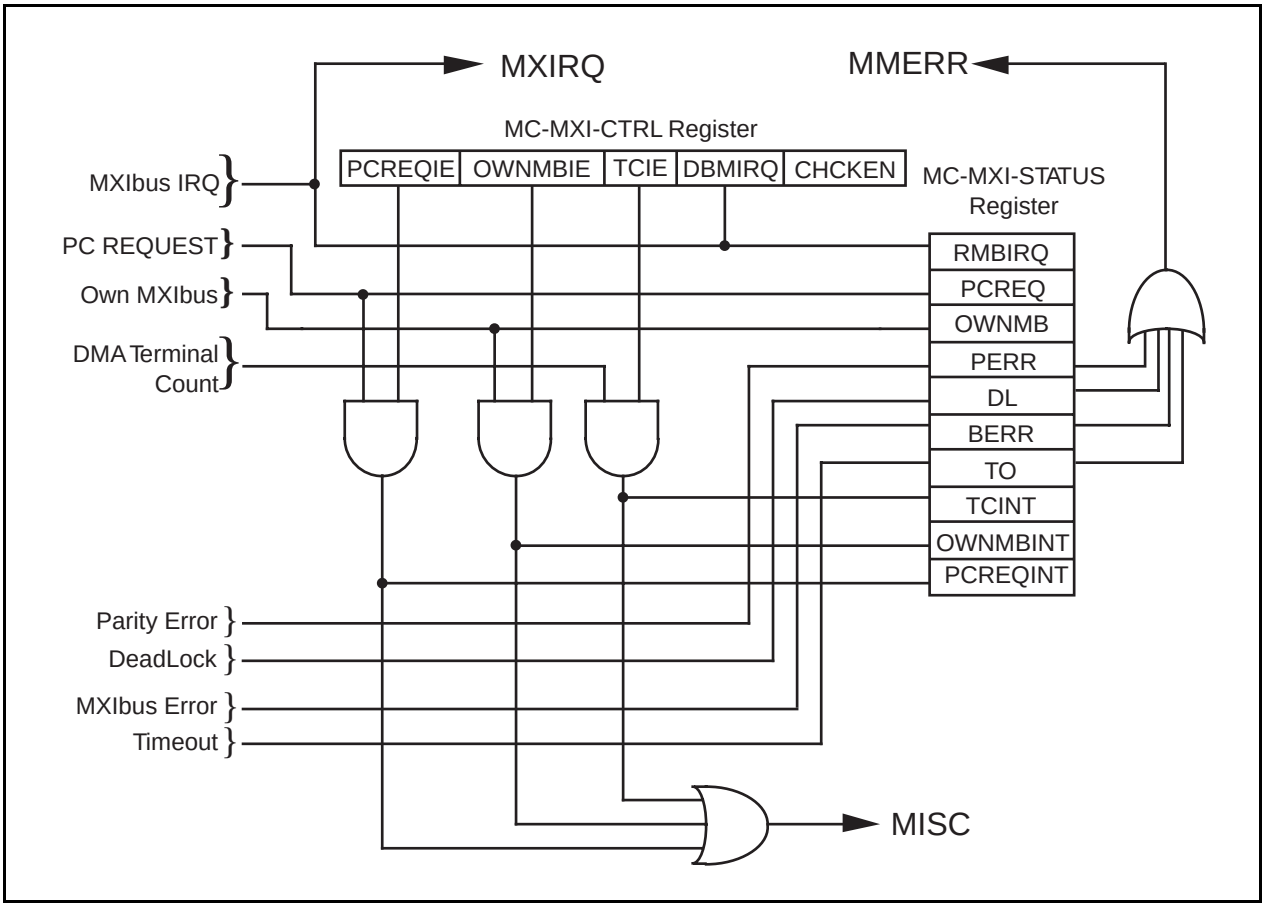


Figure 5-2. Logic to Recognize MISC, MXIRQ, and MMERR Interrupts

Recognizing MMERR Interrupts

The MC-MXI recognizes MMERR interrupts in response to the four different error conditions discussed previously. There are no enable bits to control these interrupts. However, the MC-MXI–STATUS register contains four bits; each bit reflects the state of one of the error conditions. This feature is useful for monitoring error conditions.

Recognizing MXIRQ Interrupts

The MC-MXI automatically recognizes a MXIRQ interrupt whenever the MC-MXI or some other MXIbus device asserts the MXIbus *IRQ**. The MC-MXI can generate a MXIRQ interrupt by asserting the DMBIRQ bit in the MC-MXI–CTRL register. In addition, you can monitor the state of the MXIRQ interrupt (and, in turn, the state of the MXIbus *IRQ** signal) by reading the RMBIRQ bit in the MC-MXI–STATUS register.

Generating and Handling Interrupts

A COMM, MISC, MMERR, or MXIRQ interrupt condition must generate a Micro Channel interrupt so that it can be recognized by the local CPU. Four registers are involved in this process:

- MC-MXI Interrupt Map register (POS register 4)
- Interrupt Enable register
- MC INT Status register
- MC INT Ack. register

Figure 5-3 illustrates the structure of the interrupt-generating logic.

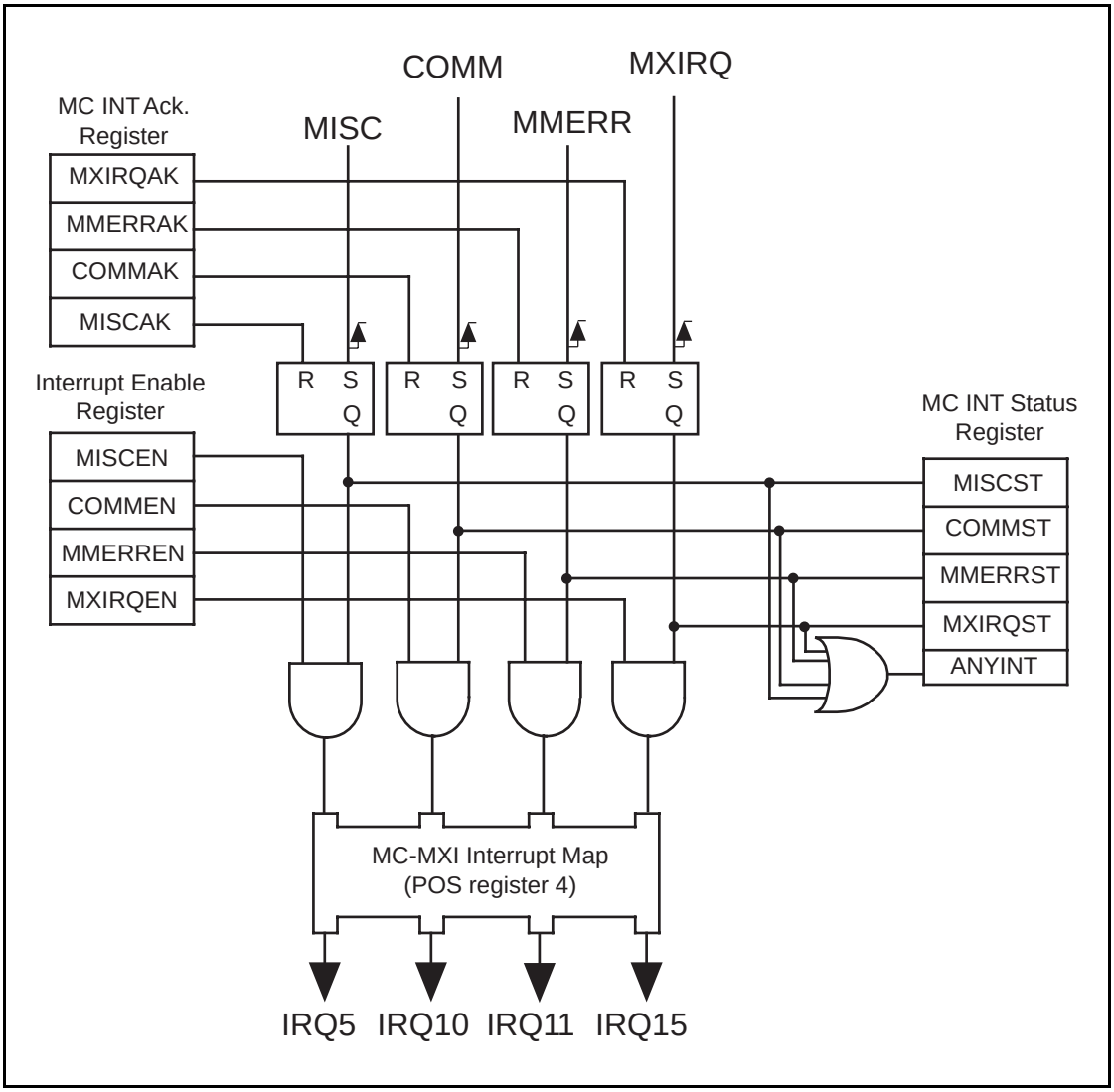


Figure 5-3. Logic to Generate Micro Channel Interrupts

The MC-MXI Interrupt Map register (POS register 4) associates the COMM, MISC, MMERR, and MXIRQ interrupts to a Micro Channel *IRQ** signal. The MC-MXI can map any or all of its four types of interrupts to either IRQ*5, IRQ*10, IRQ*11, or IRQ*15. The process of mapping the interrupts is described under the definition of the MC-MXI Interrupt Map register in Chapter 4 and in the *Initializing the MC-MXI* section, which is earlier in this chapter.

After you map the MC-MXI interrupts to a Micro Channel *IRQ** signal, you need to enable the interrupts by setting enable bits in the Interrupt Enable register. This register has one bit for each of the COMM, MISC, MMERR, and MXIRQ interrupts. The description of the Interrupt Enable register in Chapter 4 shows the arrangement of these bits.

Assuming that the MC-MXI interrupts are mapped and enabled, they should pass to the Micro Channel *IRQ** signals. If a MXIbus interrupt occurs, it is latched and remains valid until the local CPU acknowledges the interrupt. It is best to describe this process by presenting the steps involving a COMM interrupt. For the purpose of this example, assume that the COMM interrupt is mapped to Micro Channel IRQ*10:

1. A COMM interrupt is generated within the MC-MXI. This interrupt is latched on its rising edge.
2. Because the proper enable bit is set in the Interrupt Enable register, the MC-MXI asserts IRQ*10.
3. The local CPU responds to the interrupt by reading the MC INT Status register. The local CPU determines the nature of the interrupt by examining the COMMST bit in this register.
4. The local CPU acknowledges the interrupt by setting the COMMAK bit in the MC INT Ack. register.
5. The local CPU accesses the appropriate registers on the MC-MXI to remove the source of the interrupt.

The local CPU does not need to perform step 3 if the *COMM*, *MISC*, *MMERR*, and *MXIRQ* signals are mapped to different *IRQ** signals. Step 4, however, is mandatory because it clears the interrupt latch on the MC-MXI so that the MC-MXI can detect more COMM interrupts. This procedure also applies to MISC, MMERR, and MXIRQ interrupts. Refer to Chapter 4 for more details about the registers described in this example.

Byte Swapping

A problem that frequently occurs with interconnected computers is the byte ordering of data words. When a computer moves data as words, the individual bytes in these words are numerically ordered. The two methods of byte ordering are commonly referred to as *big endian* and *little endian*. Table 5-4 shows the byte orderings of 32-bit words for these different methods. Motorola 68xxx CPUs, VME, and VXI devices use the big endian approach, while Intel microprocessors use the little endian approach.

Table 5-4. Common Byte-Ordering Schemes

Big Endian			
Byte 0	Byte 1	Byte 2	Byte 3
31 24	23 16	15 8	7 0
Little Endian			
Byte 3	Byte 2	Byte 1	Byte 0
31 24	23 16	15 8	7 0

Problems can occur if a little endian computer and a big endian computer exchange byte or 16-bit word information across the data path, because the bytes will be reversed. The data can be switched back by using software, but this approach requires a great deal of processor overhead.

The MC-MXI uses hardware to automatically swap bytes. During MXIbus master-mode operations, the SWAPMB bit in the MC-MXI–CTRL register controls the byte-swapping logic. If the SWAPMB bit is 0, the bytes and 16-bit words are not swapped, but pass through the MC-MXI unchanged. If SWAPMB is set to 1, individual bytes are swapped. During MXIbus slave-mode operations, the SWAPEN bit in the Slave Configuration register controls byte swapping in the same manner as the SWAPMB bit.

Table 5-5 illustrates the operation of the byte-swapping hardware. The MC-MXI swaps byte lanes based on the size of the data transfer, which is determined by the bus signals on either the Micro Channel or the MXIbus. During 8-bit transfers, the data is always transmitted on bits 15 through 0 on the MXIbus, as required by the MXIbus specification.

Table 5-5. Byte-Swapping Operations as Function of Data Transfer Size

Byte Swapping Disabled

SWAPMB = 0	**32-Bit Data Transfer**			
Byte Lane	31–24	23–16	15–8	7–0
Micro Channel	Byte 3	Byte 2	Byte 1	Byte 0
MXIbus	Byte 3	Byte 2	Byte 1	Byte 0

SWAPMB = 0	**16-Bit Data Transfer**			
Byte Lane	31–24	23–16	15–8	7–0
Micro Channel	--	--	Byte 1	Byte 0
MXIbus	--	--	Byte 1	Byte 0

SWAPMB = 0	**8-Bit Data Transfer**			
Byte Lane	31–24	23–16	15–8	7–0
Micro Channel	--	--	--	Byte 0
MXIbus	--	--	Byte 0	--

Byte Swapping Enabled

SWAPMB = 1	**32-Bit Data Transfer**			
Byte Lane	31–24	23–16	15–8	7–0
Micro Channel	Byte 3	Byte 2	Byte 1	Byte 0
MXIbus	Byte 0	Byte 1	Byte 2	Byte 3

SWAPMB = 1	**16-Bit Data Transfer**			
Byte Lane	31–24	23–16	15–8	7–0
Micro Channel	--	--	Byte 1	Byte 0
MXIbus	--	--	Byte 0	Byte 1

SWAPMB = 1	**8-Bit Data Transfer**			
Byte Lane	31–24	23–16	15–8	7–0
Micro Channel	--	--	--	Byte 0
MXIbus	--	--	Byte 0	--

Chapter 6
Theory of Operation

This chapter contains a functional block diagram of the MC-MXI, a brief description of the major elements of the interface board, and a detailed description of both master-mode and slave-mode operation. The material in this chapter refers to many of the registers described in Chapter 4, *Register Descriptions*.

MC-MXI Architecture

The MC-MXI has several characteristics that you should consider before examining its individual components. As mentioned in previous chapters, the MC-MXI transfers data between the MXIbus and the Micro Channel. These data transfers require the MC-MXI to convert bus cycles on the MXIbus into bus cycles on the Micro Channel, and vice versa. To accomplish this task, the MC-MXI has separate bus paths for address, data, and control signals:

- The *address* path maps addresses between the Micro Channel and the MXIbus.
- The *data* path transfers information between the Micro Channel and the MXIbus and provides access to the MC-MXI internal registers.
- The *control* path influences the MC-MXI operations and behavior.

These bus paths are, for the most part, isolated from each other. For example, the data path does not have a direct connection to the address path.

By dividing information flow in this manner, the MC-MXI can execute data transfers more efficiently because it reduces data flow bottlenecks and executes some operations in parallel.

The principle of dividing the structure carries into the internal configuration of the control and address bus paths. These two bus paths have separate circuitry for data transfers from the Micro Channel to the MXIbus (master-mode operation) and from the MXIbus to the Micro Channel (slave-mode operation).

The functional block diagram of the MC-MXI shown in Figure 6-1 indicates the separate bus paths between the individual components. While there are components that require access to the address and data paths (such as the POS registers or the VXI device support block), the paths themselves never physically connect to each other. The individual components of the MC-MXI are described in detail in the following sections.

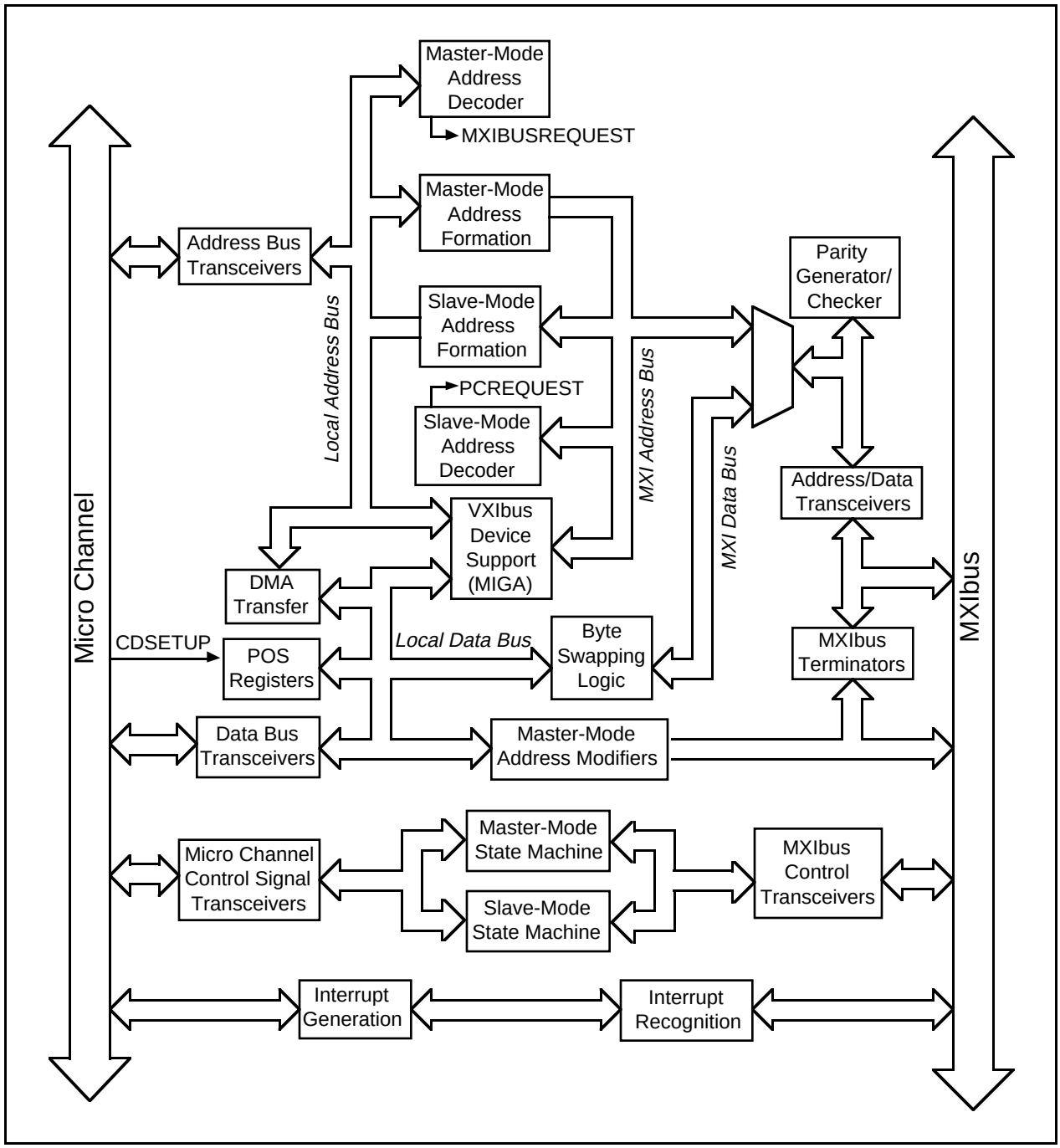


Figure 6-1. MC-MXI Architecture

MXIbus Terminators

This circuitry ensures that all MXIbus signals are properly terminated as required by the MXIbus specification. A voltage regulation circuitry provides the proper Thevenin open circuit voltage, and resistor networks provide the proper terminating characteristic impedance. You can remove the resistor networks when the MC-MXI is not an end device on the MXIbus. Because the regulation circuitry also provides TERMPWR on the MXIbus connector, you can use external terminators with the MC-MXI for easier system reconfiguration.

Slave-Mode Address Decoder

This circuitry monitors the MXIbus address modifier codes and the latched MXIbus address to determine if the current MXIbus cycle is addressing Micro Channel resources. If this is the case, the slave-mode address decoder sends a signal (*PCREQUEST*) to activate the slave-mode state machine. The slave-mode address decoder uses information from the MXI Logical Address, MXI ID, MXI Device Type, and MXI Offset registers to select the MXIbus address range(s) to decode when mapping MXIbus cycles to Micro Channel resources.

Slave-Mode Address Formation

This functional block accepts MXIbus addresses received through the MC-MXI slave-mode window, then converts them into Micro Channel addresses. This functional block uses information from the MXI Offset and WIN1AM-SMOFF registers to form the Micro Channel address. The section on slave-mode operation in Chapter 5, *Programming the MC-MXI*, explains how the MC-MXI generates Micro Channel addresses during slave-mode operations.

Slave-Mode State Machine

This state machine converts MXIbus cycles mapped through the MXIbus slave-mode window into Micro Channel cycles. It monitors the MXIbus control signals and the *PCREQUEST* signal from the slave-mode address decoder to determine when to start a MXIbus slave-mode cycle to the Micro Channel. This state machine also controls the flow of address and data information through the MC-MXI by enabling the proper transceivers at the appropriate times. For more information, refer to the *Slave-Mode Operation* section, which is located later in this chapter.

Master-Mode Address Decoder

This functional block represents the logic that monitors and decodes the latched Micro Channel address in order to determine if the Micro Channel cycle is intended for an external MXIbus device. If this is the case, this functional block sends a signal (*MBREQ*) to activate the master-mode state machine. POS register 3 controls the master-mode address decoder. The value stored in this register determines where the three master-mode windows reside in Micro Channel memory address space. The master-mode decoder also recognizes Micro Channel accesses to the MC-MXI registers.

Master-Mode Address Formation

This functional block accepts Micro Channel addresses received through the MC-MXI master-mode windows, then converts them into MXIbus addresses. This functional block uses information from the Master-Mode PAGE register to form the MXIbus address. The *MXIbus Master-Mode Operation* section in Chapter 5 explains how the MC-MXI generates MXIbus addresses during master-mode operations.

Master-Mode State Machine

This state machine converts Micro Channel cycles mapped through the master-mode windows to the MXIbus. It monitors the Micro Channel control signals and the *MXREQ* signal from the master-mode address decoder in order to determine when to start a MXIbus master-mode cycle to an external MXIbus device. This state machine also controls the flow of address and data information through the MC-MXI by enabling the proper transceivers at the appropriate times. For more information, refer to the *Master-Mode Operation* section, which is located later in this chapter.

Master-Mode Address Modifiers

This functional block consists of three registers and some support logic. The MC-MXI broadcasts the address modifiers onto the MXIbus at the beginning of a data transfer. The address modifiers affect the internal operation of the MC-MXI and tell remote MXIbus devices the type of data transfer in progress.

Parity Generator/Checker

This block represents the circuitry that generates and checks the parity on the MXIbus multiplexed address/data bus. The MC-MXI generates and checks MXIbus parity across all 32 MXIbus address/data lines.

Byte-Swapping Logic

Different computers use different byte-ordering formats when transferring data. Because the MXIbus is a processor-independent bus, different CPUs with different byte-ordering formats may attempt to communicate with each other. The MC-MXI has circuitry that can swap the order of the data bytes during multiple-byte transfers. This feature is programmable through software.

Interrupt Generation and Recognition

The MC-MXI generates internal interrupts in response to different conditions. The interrupt circuitry maps these interrupts to the Micro Channel. Two separate functional blocks make up

the MC-MXI interrupt circuitry. An interrupt generation block detects interrupt conditions and groups these conditions into four different interrupt classes. The interrupt recognition block maps the four different interrupt classes to the Micro Channel.

VXIbus Device Support

This functional block permits the MC-MXI to operate as a VXI Message-Based device. This functional block consists of registers that are required by the VXIbus specification for Message-Based devices.

Address, Data, and Control Transceivers

These logic blocks represent the interface circuitry that connects the local logic on the MC-MXI interface board to the Micro Channel. The transceivers ensure that the electrical and loading requirements of the Micro Channel bus are met. Together with the Micro Channel control lines and MC-MXI state machines, the transceivers control the enabling and direction of the Micro Channel signal flow between the MC-MXI local logic and the Micro Channel bus.

Master-Mode Operation

When the MC-MXI is operating as a MXIbus master, it converts Micro Channel memory bus cycles that are initiated by the local CPU or an alternate master on the Micro Channel bus into MXIbus cycles intended for a MXIbus slave device.

Caution: ***Never perform a MXIbus master-mode transfer through the MC-MXI that selects the MC-MXI as the slave device. Similarly, never perform a MXIbus slave-mode transfer through the MC-MXI that forces the MC-MXI to perform a MXIbus master operation.***

The above restriction stems from the fact that the MC-MXI cannot perform a MXIbus master operation and a MXIbus slave operation at the same time. The following are some examples of situations that you should avoid:

- The local CPU attempts to access the MC-MXI registers via an access to MXIbus A16 space. This scenario forces the MC-MXI to act as a master and a slave. Instead, you should access the registers from the Micro Channel I/O address space.
- The local CPU accesses a portion of the MXIbus A24 or A32 space that maps to the MC-MXI slave-mode window. In this situation, the MC-MXI is initiating a master-mode operation onto itself, thereby forcing the MC-MXI to operate as a MXIbus slave.
- A MXIbus device accesses a section of Micro Channel memory that is mapped to one of the MC-MXI master-mode windows. In this case, the MC-MXI acts as a slave in response to the MXIbus device, yet the access to the Micro Channel memory will force the MC-MXI to act as a MXIbus master.

The MC-MXI detects these types of illegal accesses by asserting a *BERR** signal on the MXIbus.

Enabling Master-Mode Operation

On reset, master-mode operation is disabled and you must program the onboard configuration registers before the MC-MXI can respond to any Micro Channel memory transfers intended for the MXIbus. To enable master mode, you need to perform the following steps:

1. Configure the master-mode options.
2. Open memory windows.
3. Program page and address modifier registers.

The *MXIbus Master-Mode Operation* section in Chapter 5 describes these steps in greater detail. After you enable master mode, any access to a master-mode window initiates a MXIbus master-mode transfer. These memory window areas remain open and enabled in Micro Channel memory space until you either reconfigure or reset the MC-MXI.

When a Micro Channel master accesses memory within a MXIbus window, the MC-MXI immediately unasserts the Micro Channel *CH CHRDY* signal in order to extend the Micro Channel cycle to match the transfer time of the MXIbus slave device.

MXIbus Arbitration

If the MC-MXI detects a Micro Channel transfer to any of the three MXIbus windows, it arbitrates for the MXIbus if it does not already own it. Arbitration begins when the MC-MXI asserts the MXIbus *BREQ** signal. The MXIbus System Controller grants the MC-MXI the MXIbus by using the *BGIN** daisy-chain signal. When the MC-MXI receives *BGIN**, it asserts the MXIbus *BUSY** signal and assumes ownership of the MXIbus. If the MC-MXI is not requesting use of the MXIbus and receives *BGIN**, it passes the bus grant to the next device via the *BGOUT** daisy-chain signal.

MXIbus devices use a Release-On-Request (ROR) arbitration scheme. This means that when the MC-MXI gains ownership of the MXIbus, it retains ownership until another MXIbus device requests the bus. Therefore, if the MC-MXI owns the MXIbus and attempts another MXIbus transfer before another MXIbus device requests the bus, the MC-MXI does not need to arbitrate again for the MXIbus. You can force the MC-MXI to hold onto the MXIbus by setting the LOCKMB bit in the MC-MXI–CTRL register. This step is useful when you are performing indivisible operations. MC-MXI block-mode and Read-Modify-Write transfers automatically hold the bus for the duration of the operation.

MXIbus Address Broadcast

As soon as the MC-MXI owns the MXIbus, it starts the transfer by broadcasting the MXIbus address and address modifiers. The MC-MXI uses the Micro Channel signal lines *S0*, S1*, A0, A1,* and *BE*(3–0)* to generate the MXIbus signals *WR*, AD0*, AD1*,* and *SIZE**. The MC-MXI also calculates the address parity and broadcasts it along with the address. The MC-MXI then waits for a prescribed address setup time and asserts the MXIbus *AS** signal to indicate to the slave device that the address on the bus is valid. After a predetermined hold time, the MC-MXI removes the address from the MXIbus *AD** bus.

The Master-Mode Address Formation logic generates a MXIbus address based on input from the PAGE register and the Micro Channel address signals. The *Memory Paging* section in Chapter 5 describes this process in more detail. How the PAGE register and the Micro Channel address combine to make a MXIbus address depends on which of the three master-mode windows is being accessed. Each window maps to some portion of MXIbus address space (either A16, A24, or A32) as determined by the master-mode address modifier values associated with each window.

Master-Mode Data Transfer

If the local CPU performs a write cycle to one of the master-mode windows, the MC-MXI transfers data from the Micro Channel onto the MXIbus *AD** bus. At the same time, the MC-MXI calculates the data parity and broadcasts it along with the data. The MC-MXI then waits for the prescribed data setup time and asserts the MXIbus *DS** signal to indicate to the slave device that the data on the bus is valid.

If the local CPU performs a read cycle from one of the master-mode windows, the MC-MXI asserts the *DS** signal after it removes the address from the *AD** bus. This action tells the participating slave that it can place its data (and data parity) on the *AD** bus.

Because the MXIbus is processor independent and does not specify the byte ordering of data, the byte-swapping circuitry on the MC-MXI ensures that data with different byte orderings pass correctly between the local CPU and the MXIbus.

Master-Mode Cycle Termination

The MC-MXI continues to assert *DS** until it receives a *DTACK** or *BERR** signal on the MXIbus. A MXIbus slave asserts the *DTACK** signal during a write transfer in order to indicate that the slave successfully received the data, or a MXIbus slave asserts the *DTACK** signal during a read transfer in order to indicate that the slave successfully placed the data on the *AD** bus.

Alternatively, a MXIbus slave or the MXIbus System Controller can terminate a data transfer by asserting *BERR**. *BERR** indicates that the transfer did not complete successfully. An addressed slave can assert *BERR** for any device-specific reason, and the MXIbus System Controller asserts *BERR** when no slave responds to the transfer within a specified amount of time. The *BERR** signal prevents the MXIbus from becoming hung during an attempt to access an absent or non-functioning MXIbus slave.

Although only a MXIbus *DTACK** or *BERR** terminates a MXIbus cycle, the corresponding Micro Channel cycle can also be terminated by a local bus timeout condition. The local timeout (controlled by the PITREADY register) becomes relevant only if MC-MXI requires more time to gain access to the MXIbus slave than the Micro Channel permits (approximately 3.5 μs).

If a local bus timeout does occur, the MC-MXI performs the following operations:

- The Micro Channel cycle terminates.
- The TO (Timeout) status bit is posted in the MC-MXI–STATUS register.
- The board interrupts the processor (optionally).

The interrupt and/or TO status bit alerts the local CPU that the transfer did not complete and must immediately be retried. The MXIbus transfer continues—even if the MC-MXI must release the Micro Channel because of the local timeout—until the MC-MXI receives a *DTACK** or *BERR** signal. The local CPU must retry the data transfer to ensure that proper data is received and that the MC-MXI circuitry remains synchronized.

The MC-MXI automatically latches the state of the Micro Channel address, control, and (in the case of a write cycle) data lines, so the MXIbus transfer can continue if the Micro Channel transfer must terminate prematurely.

Note: ***To avoid corrupting the ongoing MXIbus transfer, do not attempt any MXIbus transfer other than retrying the cycle that terminated. You can, however, continue to access the MC-MXI–STATUS registers without affecting the operation of the ongoing MXIbus transfer.***

Data Transfers Using Indivisible Operations

Indivisible cycles, such as read-modify-write operations, accommodate the use of semaphores. The MC-MXI fully supports indivisible operations, either by locking the MXIbus or by using the MXIbus indivisible access cycle. You can build true multiprocessing MXIbus systems across even multiple remote bus structures, because semaphores can exist in either the local CPU or a remote device.

During indivisible MXIbus access cycles, the MC-MXI asserts the *AS** signal over multiple cycles to ensure that other masters do not interrupt these cycles. However, because *AS** does not change state during the operation, all indivisible operations must be made to the same memory location (block-mode operation is an exception to this rule).

To initiate an indivisible data transfer, you must set the LOCKMB bit in the MC-MXI–CTRL register. By maintaining *AS** throughout the transfer, this action ensures that subsequent cycles are indivisible. Setting the LOCKMB bit also causes the MC-MXI to immediately arbitrate for the MXIbus—if it does not already own it—in anticipation of the first transfer in an indivisible sequence.

You must set the LOCKMB bit before you start an indivisible operation, and you must clear it prior to the last indivisible cycle access. Clearing the LOCKMB bit directs the MC-MXI to remove the *AS** signal at the proper time during the last cycle.

Unasserting the *AS** signal indicates to the MXIbus slave device that it can release the remote bus at the end of the current cycle. All indivisible MXIbus operations hold the remote bus (which is serviced through the MXIbus slave device) throughout the transfer in order to prevent remote bus masters from interfering with the transfer. This action ensures that indivisible MXIbus transfers are also indivisible on the remote bus.

Indivisible accesses are adequate for read-modify-write operations or multiple reads or writes to a single memory address FIFO. On the other hand, if you need to make indivisible accesses to different memory locations, you must lock the MXIbus by setting the LOCKMB bit in the MC-MXI–CTRL register.

Master-Mode Block Transfers

Block-mode MXIbus transfers are indivisible MXIbus cycles with a special address modifier code that tells the slave device to latch and keep track of the address. You can transfer blocks of consecutive memory locations at higher rates because addressing information does not have to be sent during each transfer. Because block-mode transfers work by accessing only consecutive memory locations, the MXIbus master needs to send only the initial address. The MXIbus slave is responsible for latching the initial address broadcast and generating any subsequent addresses that may be required. To send block-mode data, you can use either programmed I/O transfers or the DMA controller on the Micro Channel motherboard. Refer to the *Programming the MC-MXI for Block-Mode Transfers* section in Chapter 5 for details on how to use either method to perform block-mode data transfers.

Slave-Mode Operation

When the MC-MXI operates as a MXIbus slave, it converts MXIbus cycles initiated by a remote MXIbus master into either Micro Channel master cycles intended for a Micro Channel slave device or local access cycles intended for onboard MC-MXI registers. As a Micro Channel master, the MC-MXI arbitrates for the Micro Channel bus prior to performing the read or write access to the Micro Channel slave. A remote MXIbus master can access the onboard registers of the MC-MXI without arbitrating for the Micro Channel.

Both Micro Channel memory and I/O space are available to other MXIbus devices. However, some internal resources within the Micro Channel, such as a Micro Channel DMA controller, are not accessible via the MC-MXI because they are not connected to the Micro Channel I/O bus. Furthermore, MXIbus devices cannot access a memory location on the MC-MXI that maps back out the MXIbus on the same MC-MXI board, because the MC-MXI cannot be a MXIbus master and slave device at the same time.

Enabling Slave-Mode Operation

On a hardware reset, the MC-MXI slave-mode support is inactive. You must configure the MC-MXI to operate as a MXIbus slave. Configuring the MC-MXI for slave mode involves the following steps:

1. Open memory windows.
2. Program offset registers.
3. Configure the slave-mode options.

After you have enabled the MC-MXI for slave mode, any MXIbus access to an MC-MXI slave window initiates a Micro Channel master access or a register transfer inside the MC-MXI. This memory window remains open until you either reconfigure or reset the MC-MXI.

MC-MXI Slave-Mode Address Mapping

The MC-MXI supports two slave-mode windows. You can use one of these windows to monitor MXIbus A16 space and gain access to the MC-MXI registers. You can use the other window to monitor either A24 or A32 space and map to the Micro Channel memory space. Chapter 5 describes how to set up these two windows.

The MC-MXI interface operates as a MXIbus slave when a MXIbus cycle requests access to an area in MXIbus memory that maps to either the Micro Channel bus or the local MC-MXI registers. The MC-MXI uses the MXIbus address modifier signals AM4 and AM3 to select the number of MXIbus address lines it will decode and the space it will select. The MC-MXI maps its onboard registers into A16 space. Micro Channel memory and I/O can be mapped into either A24 or A32 space, but not to both. The M/IO* bit in the Slave Configuration register determines whether slave-mode accesses from A24 or A32 space map to the Micro Channel memory or I/O space. Table 6-1 describes the addressing modes.

Table 6-1. Micro Channel Memory Addressing Modes

Address Mode	AM4	AM3	MXIbus Address Lines	Address Space
A16	H	L	AD[15–0]	MC-MXI onboard registers
A24	H	H	AD[23–0]	Micro Channel memory or I/O (low-order 16 MB only)
A32	L	L	AD[31–0]	Micro Channel memory or I/O

Mapping A16 Space to the MC-MXI Register Set

The MC-MXI maps its configuration and communication registers into consecutive 20h bytes in the top quarter of A16 space (C000h to FFFFh). The actual address of these registers depends on the logical address of the MC-MXI. Each MXIbus device that supports the VXI register set has a unique logical address from 0h to FEh. The logical address determines the location of the device's registers in A16 space. You need to program the logical address in the Logical Address register via the Micro Channel bus before attempting any accesses to the MC-MXI onboard registers from the MXIbus. The logical address locates the configuration of the device and communication registers in A16 space according to the following formula:

Configuration Register Base Address = C000h + (Logical ID * 20h)

The MC-MXI will respond to any 8-bit or 16-bit access within this range. The six least significant MXIbus address lines select the register on the MC-MXI. Refer to the description of the Logical Address register in Chapter 4 for more information.

Micro Channel Bus Arbitration

The first step during a MXIbus slave transfer requires that the MC-MXI win control of the Micro Channel through arbitration. Upon decoding a MXIbus address that maps to Micro Channel memory or I/O space, the MC-MXI immediately requests the use of the Micro Channel. The MC-MXI competes for the Micro Channel by using the arbitration logic in the second of its two DMA channels (the first DMA channel operates when the MC-MXI is a MXIbus master).

The MC-MXI can hold the Micro Channel as long as it needs to, provided that another bus master does not request the channel. If another bus master requests access to the Micro Channel, the MC-MXI must relinquish the Micro Channel within 7.8 µs or when it completes its current data transfer (whichever time is shorter). The PITLOCAL timer ensures that the MC-MXI relinquishes the Micro Channel within 7.8 µs.

Slave-Mode Data Transfer

When the MC-MXI wins control of the Micro Channel bus, it decodes the MXIbus signal lines *AM4**, *AM3**, *WR**, *SIZE**, *AD1**, and *AD0** to determine the type of transfer. Transfer types that the MC-MXI supports are 8-bit or 16-bit memory reads and writes and 8-bit or 16-bit I/O reads and writes. The MC-MXI does not discriminate between privileged and nonprivileged or data and program access modes.

Slave-Mode Cycle Termination

The MC-MXI terminates the Micro Channel cycle by unasserting the Micro Channel cycle control signals. The Micro Channel slave that the MC-MXI was accessing responds by unasserting *IOCHRDY*. The MC-MXI then terminates the MXIbus access: the MC-MXI asserts the MXIbus *DTACK** signal if the cycle completed successfully; the MC-MXI asserts the *BERR** signal if the cycle completed with an error. If the Micro Channel bus is not locked and the MXIbus master is not performing block-mode transfers to the Micro Channel, the MC-MXI also releases the Micro Channel. The Micro Channel is not released after a transfer if the LOCKMC bit in the Slave Configuration register is set to 1 or if there is an ongoing block-mode transfer.

The MC-MXI automatically terminates a block-mode transfer and releases the Micro Channel bus if it detects an error, but the MC-MXI does not automatically clear the LOCKMC bit. You must either set the LOCKMC bit back to 1 in the Slave Configuration register or generate a hard reset to take the MC-MXI out of lock mode.

Five conditions can cause the MC-MXI to terminate a MXIbus slave-mode access to Micro Channel memory or I/O. The first condition is a successful MXIbus slave-mode cycle. The other four conditions are error conditions. The MC-MXI terminates successful data cycles by

asserting the *DTACK** signal, and it terminates unsuccessful data cycles by asserting the *BERR** signal. The possible conditions for terminating a data transfer are as follows:

1. No errors occurred and the cycle completed within the specified MXIbus System Controller timeout period. The MC-MXI terminates the current cycle with a *DTACK**.

2. A parity error occurred when either the address or data was broadcast on the MXIbus. The MC-MXI terminates the current cycle with a *BERR**, and the MC-MXI will not perform the Micro Channel cycle.

3. An error occurred on the Micro Channel bus. The Micro Channel *IOCHCK* signal indicates whether an error occurred on the Micro Channel. If the MC-MXI detects that *IOCHCK* is asserted at any time during the MXIbus slave access, it terminates the access by asserting *BERR**. Notice that there is no guarantee that the transfer actually took place or that the data was correctly stored or retrieved. Asserting *IOCHCK* indicates a serious error and usually the signal maps to a non-maskable interrupt on the local CPU.

4. A block-mode transfer extended beyond the maximum memory address (FFFFFFh) of the Micro Channel. The MC-MXI terminates the block-mode cycle with *BERR**. The Micro Channel cycle that caused the error is not performed.

5. A MXIbus device attempts to write to the MC-MXI Signal register FIFO when the FIFO is full. The MC-MXI terminates the current cycle by asserting *BERR**.

The MXIbus System Controller can also terminate a MXIbus cycle if the cycle exceeds the MXIbus System Controller timeout period.

Slave-Mode Block Transfers

The MC-MXI has an onboard address generator to accommodate block-mode slave transfers to the Micro Channel. During slave-mode block accesses, the MC-MXI latches the initial address into an onboard counter. The counter then supplies all the subsequent addresses throughout the block-mode transfer. The MC-MXI automatically holds and locks the Micro Channel during block-mode slave accesses until after the last transfer of the block mode completes.

The MC-MXI supports block-mode slave transfers to both Micro Channel memory and I/O space. The MC-MXI checks the value of the address generator during each transfer to ensure that the address remains within the range of the slave-mode window. The address generator increments by 1, 2, or 4 bytes, depending on whether the block-mode transfer is a byte, word, or longword, respectively.

Reset Circuitry on the MC-MXI

The MC-MXI implements resets as a hierarchy of different levels, making it possible for the local CPU to control the effect of a reset signal. The MC-MXI reset levels are as follows:

- *Soft resets* have the smallest scope of all resets. You can trigger a soft reset by setting the RESET bit in the MXI Control register. A soft reset affects the contents of the MC-MXI registers between 0 and 31 (hex) from the MC-MXI I/O base address. You must enable the soft reset prior to triggering it by setting the SRSTEN bit in the Communication Control register. After triggering a soft reset, you clear it by setting the RESET bit to 0 and by performing a read to the Soft Reset Service register.

- *Hard resets* are triggered by a channel reset or by setting the HRDRESET bit in the WAITSTATES register. A hard reset affects all the MC-MXI hardware (including the registers affected by a soft reset). However, the POS registers and the registers between addresses 140 and 145 (hex), which are above the MC-MXI base I/O address, are not affected.

- *Channel resets* are triggered by asserting the channel reset signal on the Micro Channel. This type of reset affects all the registers and circuitry on the MC-MXI, including the registers not covered by the hardware reset.

The register descriptions in Chapter 4 describe how these different resets affect bit fields within individual registers.

Appendix A
Specifications

This appendix lists various MC-MXI module specifications, such as physical dimensions and power requirements.

Capability Codes

MXIbus

Capability Code	Description
MA32	Master Mode A32, A24, and A16 addressing
MBLT	Master Mode block transfers
SA32	Slave Mode A32, A24, and A16 addressing
SBLT	Slave Mode block transfers
MD32	Master Mode D32, D16, and D08 data sizes
SD32	Slave Mode D32, D16, and D08 data sizes
SC	Optional MXIbus System Controller
FAIR	Can be a fair MXIbus requester
LOCK	Can lock the MXIbus for indivisible transfers
TERM	Can terminate the MXIbus

Micro Channel

Capability Code	Description
MCMD32	Master D32, D16, and D08 data sizes
MCSD32	Slave D32, D16, and D08 data sizes
MCFAIR	Can be a fair Micro Channel requester
AM	Can function as an MC alternate master
LOCK	Can lock the Micro Channel for indivisible transfers
DMA32	Supports D32, D16, or D08 DMA transfers
INT	Can be a Micro Channel Interrupter

Electrical

Source	Typical	Direct Current (max)
+5 VDC	3.0 A	3.5 A

Environmental

Component Temperature	0° to 70° C (32° to 158° F) operating -55° to 150° C (-67° to 302° F) storage
Emissions	FCC Class A
Relative Humidity	0% to 95% noncondensing; operating 0% to 100% noncondensing; storage
Safety	Not applicable
Shock and Vibration	Not applicable

Physical

Board Size	Standard full-length Micro Channel board 29.21 mm by 8.82 mm (11.5 in. by 3.475 in.)
Connectors	Single fully implemented MXIbus connector
Slot Requirements	Single 32-bit Micro Channel slot

Reliability

MTBF	Contact Factory

Requirements

Memory space required	64 KB or 32,832 KB
I/O space required	4 KB

Timing

Master Mode

Transfer Type	Transfer Rate
Write	530 ns
Read	430 ns
Block Write	290 ns
Block Read	190 ns

Slave Mode

Transfer Type	Transfer Rate
Write	340 ns
Read	440 ns
Block Write	310 ns
Block Read	360 ns

Other

Daisy-Chain Delay 120 ns
(Passing GIN to GOUT or GOUT generation from System Controller)

Appendix B
Mnemonics Key

This appendix contains an alphabetical listing of the mnemonics that this manual uses to describe signals, registers, and register bits. Refer also to the *Glossary*.

The mnemonic types in the key that follows are abbreviated to mean the following:

B	Register Bit
MBS	MXIbus Signal
MCS	Micro Channel Signal
R	Register
S	MC-MXI Internal Signal

(R) or (W) following a bit or register description signifies that the register or bit is either read-only or write-only, respectively. (R/W) indicates that it has both capabilities.

Mnemonic	Type	Definition
A		
A[31-0]	B	A32 Pointer Address Offset Bits (R/W)
A[23-0]	B	A24 Pointer Address Offset Bits (R/W)
AD[31-0]*	MBS	MXIbus Address/Data Bus
ADSP[1-0]	B	Address Space Field (R/W)
AM[4-0]*	MBS	MXIbus Address Modifier Lines
AND/OR DMA1	B	And or OR Software and Hardware DMA Requests Bit
AND/OR DMA2	B	And or OR Software and Hardware DMA Requests Bit
ANYINT	B	Any Interrupt Bit
ARB1[3-0]	B	DMA Channel 1 Arbitration Level Bits
ARB2[3-0]	B	DMA Channel 2 Arbitration Level Bits
AS*	MBS	MXIbus Address Strobe Signal
B		
BCD	B	Binary or Decimal Counting Bit
BE*	MCS	Micro Channel signal
BERR*	MBS	MXIbus Bus Error Signal
BERR	B	Bus Error Bit (R)
BGIN*	MCS	MXIbus Grant In
BGOUT*	MCS	MXIbus Grant Out
BLOCK	B	Block Mode Master Transfer Bit
BREQ*	MBS	MXIbus Request Signal
BRW	B	Byte order for timer read/write
BURST1EN	B	Burst Enable for DMA Channel 1 Bit
BURST2EN	B	Burst Enable for DMA Channel 2 Bit
BUSY*	MBS	MXIbus Bus Busy Signal
C		
CARDEN	B	Card Enable Bit
CDSETUP	S	Micro Channel Card Setup
CHCK*	B	Channel Check Bit
CHCK*	S	MCS signal indicating a severe Micro Channel error
CHCKEN	B	I/O Channel Check Interrupt Enable Bit
CHCKSTAT	B	Channel Check Status Bit
CH CHRDY	MCS	Micro Channel Ready
CLASS[1-0]	B	Device Class Field (R/W)
CMDR*	B	MXIbus Commander Bit (R/W)
COMM[1-0]	B	Communications Interrupt Map Bits
COMMAK	B	Communication Interrupt Acknowledge Bit
COMMEN	B	Communication Interrupt Enable Bit
COMMST	B	Communication Interrupt Status Bit

Mnemonic	Type	Definition
D		
D[47-32]	B	Data Bits for Data Extended (In) Register (R/W)
D[31-16]	B	Data Bits for Data High Register (R/W)
D[15-0]	B	Data Bits for Data Low Register (R/W)
DIR	B	Data In Ready Bit (R/W)
DL	B	Deadlock Bit (R)
DL[1-0]	B	Data Length (R)
DMA1EN	B	DMA Channel 1 Enable Bit
DMA2EN	B	DMA Channel 2 Enable Bit
DMBIRQ	B	Drive MXIbus Interrupt Request Bit (W)
DOR	B	Data Out Ready Bit (R/W)
DS*	MBS	MXIbus Data Strobe Signal
DTACK*	MBS	Data Transfer Acknowledge Signal
E		
ERR*	B	Error Bit (R/W)
F		
FAIR	B	Fairness Bit
FAIRMB	B	Fair MXIbus Requester Bit (W)
FHS*	B	MXIbus Fast Handshake Bit (R)
FHSACT*	B	Fast Handshake Active Bit (R)
FIFO		First In First Out Data Buffer
G		
GIN*	MBS	MXIbus Bus Grant In Signal
GOUT*	MBS	MXIbus Bus Grant Out Signal
H		
HRDRESET	B	Hard Reset Bit (R/W)
I		
I/O[3–0]	B	Input/Output Base Address Bits
INTACK	B	Acknowledge Bit
INTRPTR	B	MXIbus Interrupter Bit (R/W)
IOW[2-0]	B	I/O Wait States Bits
IRQ*	MBS	MXIbus Interrupt Request Signal

Mnemonic	Type	Definition
L		
LA[7-0]	B	Logical Address Field (R/W)
LAREN	B	Logical Address Register Enable Bit (W)
LARENRD	B	Logical Address Register Enable Bit (R)
LARENWR	B	Logical Address Register Enable Bit (W)
LBARB	B	Local Bus ArBiter Enable Bit
LOCKED*	B	MXIbus Servant Lock Bit (R/W)
LOCKMB	B	Lock MXIbus Bit (W)
LOCKMC	B	Lock Micro Channel
M		
M[2-0]	B	Operating Mode Bits
M/IO*	B	Memory/IO Select
MANID[11-0]	B	Manufacturer ID Field (R/W)
MASTER*	B	MXIbus Master Bit (R/W)
MBBLOCKEN	B	MXIbus Block Mode Enable Bit
MBREQ	B	MXIbus Request Bit
MBSC	B	MXIbus System Controller Bit (W)
MDL[11-0]	B	Model Code Field (R/W)
MDRQEN	B	Master Mode DMA Transfer Enable Bit
MEM[3-0]	B	Memory Base Address Bits
MEMACT	B	Memory Active Bit (W)
MEMACTW	B	Memory Active Bit
MIGA		Message-Based Interface Gate Array
MISC		Miscellaneous interrupts
MISCAK	B	General Interrupt Acknowledge Bit
MISCEN	B	General Interrupt Enable Bit
MISCINT[1-0]	B	General Interrupts Bits
MISCST	B	General Interrupt Status Bit
MMEN	B	Master Mode Enable Bit
MMERR[1-0]	B	Error Condition Bits
MMERRAK	B	Error Condition Interrupt Acknowledge Bit
MMERREN	B	Error Condition Interrupt Enable Bit
MMERRST	B	Error Condition Interrupt Status Bit
MQEDET	B	Multiple Query Error Detect Bit (R)
MW[2-0]	B	Memory Wait States Bits
MXIRQ[1-0]	B	MXIbus Interrupt Map Bits
MXIRQAK	B	Acknowledge Interrupts from the MXIbus IRQ* Signal Bit
MXIRQEN	B	MXIbus Interrupt Enable Bit
MXIRQST	B	MXIbus IRQ* Interrupt Status Bit
O		
OFF[15-0]	B	MXIbus A24 or A32 Offset Bits (R/W)
OWNMB	B	MXIbus Ownership Bit (R)

Mnemonic	Type	Definition
OWNMBIE	B	MXIbus Ownership Interrupt Enable Bit
OWNMBINT	B	MXIbus Ownership Interrupt Bit

P

PAR*	MBS	MXIbus Parity Signal
PASSED	B	Device Passed Bit (R/W)
PCREQ	B	PC Request Bit
PCREQIE	B	PC Request Interrupt Enable Bit
PCREQINT	B	PC Request Interrupt Bit
PCREQINT	S	PC Request Interrupt Signal
PCREQUEST	MCS	PC Request Signal
PERR	B	Parity Error Bit (R)
PITSC	R	System Controller PIT
POS1026	B	Special Mode Enable Bit
PREEMPT*	S	Micro Channel Bus Preemption Signal
PROT[9-4]	B	Protocol Register Bits [9-4] (R/W)
PROT[3-0]	B	Protocol Register Bits [3-0] (R/W)

R

RDERR	B	Read Error Bit (R/W)
RDIE	B	Read Data Interrupt Enable Bit (W)
RDIERD	B	Read Data Interrupt Enable Bit (R)
RDIEST	B	Read Data Interrupt Enable Status Bit (R)
RDIEWR	B	Read Data Interrupt Enable Bit (W)
RDRDY	B	Read Ready Bit (R)
RDREQ	B	Read Data Request Bit (R)
READY	B	Device Ready Bit (R/W)
RESET	B	Soft Reset Bit (W)
RESETSRC	B	Reset Source Bit
RESP[1-0]	B	Response Register Bits [1-0] (R/W)
RESP[14]	B	Response Register Bit [14] (R/W)
RMBIRQ	B	MXIbus Interrupt Request Bit (R)
RQMEM[3-0]	B	Required Memory Field (R/W)

S

S*[1-0]	MCS	Micro Channel Status
S/W DMA1 REQUEST	B	Software DMA Request on Channel 1
S/W DMA2 REQUEST	B	Software DMA Request on Channel 2
SC[1-0]	B	Select Clock
SCCLK	R	System Controller Clock Period
SCCLK[7–0]	B	System Controller Clock Period
SHMEM*	B	MXIbus Shared Memory Bit (R/W)
SIG[15-0]	B	MXIbus Signal FIFO (R/W)
SIGIE	B	Signal Interrupt Enable Bit (W)

Mnemonic	Type	Definition
SIGIERD	B	Signal Interrupt Enable Bit (R)
SIGIEWR	B	Signal Interrupt Enable Bit (W)
SIGREG*	B	MXIbus Signal Register Bit (R/W)
SIGREQ	B	Signal Request Bit (R)
SIZE*	MBS	MXIbus Size Signal
SM24OFF[3-0]	B	Slave Mode A24 Offset Bits
SM32OFF[3-0]	B	Slave Mode A32 Offset Bits
SRSTEN	B	Soft Reset Enable Bit (W)
SRSTENRD	B	Soft Reset Enable Bit (R)
SRSTENWR	B	Soft Reset Enable Bit (W)
SRSTIE	B	Soft Reset Interrupt Enable Bit (W)
SRSTIERD	B	Soft Reset Interrupt Enable Bit (R)
SRSTIEWR	B	Soft Reset Interrupt Enable Bit (W)
SRSTREQ	B	Soft Reset Request Bit (R)
STAT[12-4]	B	MXI Status Register Bits [12-4] (R/W)
SWAPEN	B	Enable Byte Swapping
SWAPMB	B	Multi-Byte Swap Bit
SYSCLK	S	System Controller Base Clock
SYSCLKEN	B	System Controller Timeout Enable Bit
SYSCLKINTEN	B	System Controller Base Clock Interrupt Enable

T

T[7-0]	B	System Controller Clock Period Bits
TCENDEN	B	Terminal Count Enable Bit
TCIE	B	Terminal Count Interrupt Enable Bit
TCINT	B	Terminal Count Interrupt Bit
TCINT	S	Terminal Count Interrupt Signal
TERMPWR	MBS	MXIbus Termination Power Line
TIMERAK	B	Timer Interrupt Acknowledge Bit
TIMERTST	B	Special Mode for System Controller Timeout Bit
TO	B	Timeout Bit (R)

W

W1AM[4-0]	B	Window 1 Address Modifier Bits
W2AM[4-0]	B	Window 2 Address Modifier Bits
W3AM[4-0]	B	Window 3 Address Modifier Bits
WR*	MBS	MXIbus Write Signal
WRERR	B	Write Error Bit (R/W)
WRIE	B	Write Data Interrupt Enable Bit (W)
WRIERD	B	Write Data Interrupt Enable Bit (R)
WRIEWR	B	Write Data Interrupt Enable Bit (W)
WRRDY	B	Write Ready Bit (R/W)
WRREQ	B	Write Data Request Bit (R)

Appendix C
MXIbus Connector Description

This appendix describes the connector pin assignments for the MXIbus connector.

MXIbus Connector

The MXIbus signals are assigned to the device connector, as shown in Figure C-1 and Table C-1.

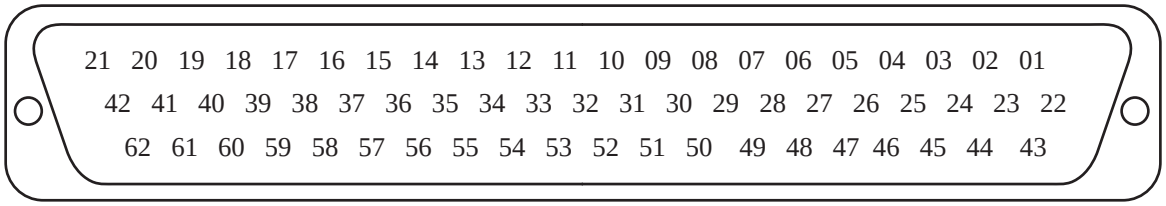


Figure C-1. MXIbus Connector

Table C-1. MXIbus Connector Signal Assignments

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	*AM4**	22	*AD15**	43	*PAR**
2	*AM3**	23	*AD14**	44	*SIZE**
3	*AM2**	24	*AD13**	45	*BREQ**
4	*AM1**	25	*AD12**	46	*BUSY**
5	*AM0**	26	*AD11**	47	*GND*
6	*AD31**	27	*AD10**	48	*GND*
7	*AD30**	28	*AD09**	49	*GND*
8	*AD29**	29	*AD08**	50	*GND*
9	*AD28**	30	*AD07**	51	*GND*
10	*AD27**	31	*AD06**	52	*GND*
11	*AD26**	32	*AD05**	53	*GND*
12	*AD25**	33	*AD04**	54	*GND*
13	*AD24**	34	*AD03**	55	*GND*
14	*AD23**	35	*AD02**	56	*GND*
15	*AD22**	36	*AD01**	57	*GND*
16	*AD21**	37	*AD00**	58	*GND*
17	*AD20**	38	*DS**	59	*GOUT**
18	*AD19**	39	*AS**	60	*GIN**
19	*AD18**	40	*WR**	61	*IRQ**
20	*AD17**	41	*DTACK**	62	*TERMPWR*
21	*AD16**	42	*BERR**		

The MXIbus is a flexible cable with 62 pin connectors at each end. It is comprised of 49 active signals, 12 ground lines, and 1 line for terminator power. Each signal is individually twisted with a ground and encased in shield foil. Signal names ending with an asterisk (*) are active low. Table C-2 describes the signals on the MXIbus connector and groups them in five categories.

Table C-2. MXIbus Signal Groupings

Category	Description	Signal Name	Lines	Pin Numbers
Address/Data	Address/Data	*AD[31–00]**	32	6–37
	Address Modifier	*AM[4–0]**	5	1–5
	Address Strobe	*AS**	1	39
	Transfer Size	*SIZE**	1	44
	Read/Write	*WR**	1	40
	Data Strobe	*DS**	1	38
	Data Acknowledge	*DTACK**	1	41
	Parity	*PAR**	1	43
Arbitration	MXIbus Busy	*BUSY**	1	46
	MXIbus Request	*BREQ**	1	45
	MXIbus Grant In	*GIN**	1	60
	MXIbus Grant Out	*GOUT**	1	59
Interrupt	Interrupt Request	*IRQ**	1	61
Utility	MXIbus Error	*BERR**	1	42
Power	Ground	*GND*	12	47–58
	Terminator Power	*TERMPWR*	1	62

Notice that there are 12 ground contacts on the connector. The 48 twisted ground lines in the cable are generated from these lines under the cable connector hood. Also notice that although there are two connector contacts required for the *GIN**-*GOUT** daisy-chain, only one signal line is required in the cable assembly.

For more information, refer to the MXIbus specification.

Appendix D
Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve technical problems you might have as well as a form you can use to comment on the product documentation. Filling out a copy of the *Technical Support Form* before contacting National Instruments helps us help you better and faster.

National Instruments provides comprehensive technical assistance around the world. In the U.S. and Canada, applications engineers are available Monday through Friday from 8:00 a.m. to 6:00 p.m. (central time). In other countries, contact the nearest branch office. You may fax questions to us at any time.

Corporate Headquarters
(512) 795-8248
Technical support fax: (800) 328-2203
(512) 794-5678

Branch Offices	**Phone Number**	**Fax Number**
Australia	(03) 879 9422	(03) 879 9179
Austria	(0662) 435986	(0662) 437010-19
Belgium	02/757.00.20	02/757.03.11
Denmark	45 76 26 00	45 76 71 11
Finland	(90) 527 2321	(90) 502 2930
France	(1) 48 14 24 00	(1) 48 14 24 14
Germany	089/741 31 30	089/714 60 35
Italy	02/48301892	02/48301915
Japan	(03) 3788-1921	(03) 3788-1923
Netherlands	03480-33466	03480-30673
Norway	32-848400	32-848600
Spain	(91) 640 0085	(91) 640 0533
Sweden	08-730 49 70	08-730 43 70
Switzerland	056/20 51 51	056/20 51 55
U.K.	0635 523545	0635 523154

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Use additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___)_____ Phone (___)_____

Computer brand _____ Model _____ Processor _____

Operating system _____

Speed _____MHz RAM _____MB Display adapter _____

Mouse _____yes _____no Other adapters installed_

Hard disk capacity _____MB Brand _____

Instruments used _____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is _____

_____

_____

_____

_____

List any error messages_

_____

_____

_____

The following steps will reproduce the problem _____

_____

_____

_____

_____

MC-MXI Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

- MC-MXI Logical Address _____
- MC-MXI Device Type _____
- MC-MXI Address Space _____
- MC-MXI Shared Memory Size _____
- MC-MXI Shared Memory Buffer _____
- MC-MXI Resource Manager Delay _____
- MC-MXI Servant Area Size _____
- MC-MXI Protocol Register _____
- Number of MC-MXI Handlers _____
- Number of MC-MXI Interrupters _____
- MXIbus Fair Requester? _____
- MXIbus Terminators Installed or Removed? _____
- MC-MXI Memory Base Window Address _____
- MC-MXI Base I/O Address _____
- MC-MXI Interrupt Level(s) _____
- MC-MXI Arbitration Level(s) _____
- MC-MXI Hardware Revision _____
- Application Programming Language _____
- Programming Language Interface Revision _____

Other Products

- Computer Make and Model _____
- Microprocessor _____
- Clock Frequency (Bus and Microprocessor) _____
- Total Memory in System _____
- Type of Video Board Installed _____
- Operating System and Version _____
- Programming Language Version _____
- Other Boards in System _____
- Interrupt Level of Other Boards _____
- Other Devices in System _____
- Static Logical Addresses of Other Devices _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: **MC-MXI User Manual**

Edition Date: **May 1994**

Part Number: **320297-01**

Please comment on the completeness, clarity, and organization of the manual.

If you find errors in the manual, please record the page numbers and describe the errors.

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

Phone (_____) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway, MS 53-02
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
MS 53-02
(512) 794-5678

Glossary

Prefix	Meaning	Value
μ-	micro-	10^{-6}
m-	milli-	10^{-3}
K-	kilo-	10^{3}
M-	mega-	10^{6}
G-	giga-	10^{9}

Symbols

°	degrees
%	percent
Ω	ohms

A

A	amperes
A16 space	MXIbus and VXIbus address space equivalent to the VME 64 KB *short* address space. In VXI, the upper 16 KB of A16 space is allocated for use by VXI devices' configuration registers. This 16 KB region is referred to as VXI configuration space.
A24 space	MXIbus and VXIbus address space equivalent to the VME 16 MB *standard* address space.
A32 space	MXIbus and VXIbus address space equivalent to the VME 4 GB *extended* address space.
ADF	Adapter Descriptor File

B

B	bytes
backplane	An assembly, typically a printed circuit board, with 96-pin connectors and signal paths that bus the connector pins.

block-mode transfer	An uninterrupted transfer of data elements in which the master sources only the first address at the beginning of the cycle. The slave is then responsible for incrementing the address on subsequent transfers so that the next element is transferred to or from the proper storage location.
bus master	A device that can request the Data Transfer Bus (DTB) for the purpose of accessing a slave device.
byte order	How bytes are arranged within a word or how words are arranged within a longword. Motorola ordering stores the most significant (MSB) byte or word first, followed by the least significant byte (LSB) or word. Intel ordering stores the LSB or word first, followed by the MSB or word.

C

C	Celsius
CMOS	complementary metal-oxide semiconductor
Commander	A Message-Based device that is also a bus master and can control one or more Servants.
CPU	central processing unit

D

DMA	direct memory access
DOS	disk operating system
DRAM	Dynamic RAM (Random Access Memory): storage that the computer must refresh at frequent intervals.
dynamic configuration	A method of automatically assigning logical addresses to VXIbus devices at system startup or other configuration times.
dynamically configured device	A device that has its logical address assigned by the Resource Manager. A VXI device initially responds at Logical Address 255 when its MODID line is asserted. A MXIbus device responds at Logical Address 255 during a priority select cycle. The Resource Manager subsequently assigns it a new logical address, which the device responds to until powered down.

E

embedded controller	An intelligent CPU (controller) interface plugged directly into the VXI backplane, giving it direct access to the VXIbus. It must have all of its required VXI interface capabilities built in.

EMI	electromagnetic interference
extended controller	A mainframe extender with additional VXIbus controller capabilities.
Extended Longword Serial	A form of Word Serial communication in which Commanders and Servants communicate with 48-bit data transfers.
external controller	In this configuration, a plug-in interface board in a computer is connected to the VXI mainframe via one or more VXIbus extended controllers. The computer then exerts overall control over VXIbus system operations.

F

F	Fahrenheit
fair requester	A MXIbus master that will not arbitrate for the MXIbus after releasing it until it detects the bus request signal inactive. This ensures that all requesting devices will be granted use of the bus.
FCC	Federal Communications Commission

G

GB	gigabytes
GND	Electrical Ground
GPIB	General Purpose Interface Bus: the industry-standard IEEE 488 bus.

H

Hz	hertz: events per second.

I

I/O	input/output: the techniques, media, and devices used to achieve communication between entities.
IC	integrated circuit
in.	inches
interrupt	A means for a device to request service from another device.
interrupt handler	A VMEbus functional module that detects interrupt requests generated by Interrupters and responds to those requests by requesting status and identify information.

K

KB	1,024 or 2^{10}: bytes of memory

L

Longword Serial	A form of Word Serial communication in which Commanders and Servants communicate with 32-bit data transfers instead of 16-bit data transfers as in the normal Word Serial Protocol.

M

m	meters
MB	1,048,576 or 2^{20}: bytes of memory
master	A functional part of an MXI/VME/VXIbus device that initiates data transfers on the backplane. A transfer can be either a read or a write.
Message-Based device	An intelligent device that implements the defined VXIbus registers and communication protocols. These devices are able to use Word Serial Protocol to communicate with one another through communication registers.
MHz	megahertz
MIGA	Message-Based Interface Gate Array: a proprietary National Instruments chip.
mm	millimeters
ms	milliseconds
MXIbus	Multisystem Extension Interface Bus

N

NI-VXI	The National Instruments bus interface software for VME/VXIbus and MXIbus systems.
ns	nanoseconds
nonprivileged access	One of the defined types of MXIbus or VMEbus data transfers: indicated by certain address modifier codes. Each of the defined VMEbus address spaces has a defined nonprivileged access code.

P

PC AT	Personal Computer Advanced Technology
PIT	Peripheral Interface Timer
POS	Programmable Option Select
privileged access	See *supervisory access*.

R

RAM	random-access memory
Register-Based device	A Servant-only device that supports VXIbus configuration registers. Register-Based devices are typically controlled by Message-Based devices via device-dependent register reads and writes.
Resource Manager	A Message-Based Commander located at Logical Address 0. It provides configuration management services, such as address map configuration, Commander and Servant mappings, and self-test and diagnostic management.
ROM	read-only memory
RONR	Request On No Request
ROR	Release On Request: a type of MXIbus or VMEbus arbitration where the current VMEbus master relinquishes control of the bus only when another bus master requests the VMEbus.

S

s	seconds
Servant	A device controlled by a Commander.
statically configured device	A device whose logical address cannot be set through software; that is, it is not dynamically configurable.
supervisory access	One of the defined types of MXIbus or VMEbus data transfers: indicated by certain address modifier codes.
SYSCLK	System Clock Driver: a VMEbus functional module that provides a 16 MHz timing signal on the utility bus.
SYSRESET	System Reset Driver: asserts a signal to indicate a system reset or power-up condition.
System Controller	A functional module that has arbiter, daisy-chain driver, and MXIbus cycle timeout responsibility.

T

TC	terminal count
trigger	Either TTL or ECL lines used for intermodule communication.
tristated	Defines logic that can have one of three states: low, high, and high impedance.

V

V	volts
VDC	volts direct current
VIs	virtual instruments
VME	Versa Module Eurocard or IEEE 1014
VXI	VME eXtensions for Instrumentation (bus)
VXIbus	VMEbus Extensions for Instrumentation

W

W	watts
Word Serial Protocol	The simplest required communication protocol supported by Message-Based devices in the VXIbus system. It utilizes the A16 communication registers to transfer data by using a simple polling handshake method.

Index

A

A16 address space
  definition, 1-11
  mapping to MC-MXI register set, 6-10
  privileges, 1-11
A[23-16] bits, 4-46
A24 address space
  definition, 1-11
  privileges, 1-11
A24 Pointer Register, 4-25 to 4-26
  A24 Pointer (High), 4-25
  A24 Pointer (Low), 4-26
A[31-24] bits, 4-46
A32 address space
  definition, 1-11
  privileges, 1-11
A32 Pointer Register, 4-27
  A32 Pointer (High), 4-27
  A32 Pointer (Low), 4-27
AD[31-0]* signal
  basic data transfers, 1-12
  block-mode data transfers, 1-13
  definition, 1-7
  priority select data transfers, 1-14
address broadcast, MXIbus, 6-6 to 6-7
address decoder
  master-mode operation, 6-3
  slave-mode operation, 6-3
address formation
  master-mode operation, 6-4
  slave-mode operation, 6-3
  using Master-Mode PAGE register (table), 5-8
address mapping, slave-mode
  mapping A16 space to MC-MXI register set, 6-10
  Micro Channel memory addressing modes (table), 6-10
  theory of operation, 6-10
address modifiers
  codes (table), 1-10
  master-mode operation, 6-4
address path, 6-1
address transceivers, 6-5
addressing modes and address space, 1-10 to 1-11
  A16 space, 1-11
  A24 space, 1-11
  A32 space, 1-11
  address modifier codes (table), 1-10
  privileges and modes, 1-11
ADSP[1-0] bit, 4-7
AM[4-0]* signal
  basic data transfers, 1-12
  block-mode data transfers, 1-13
  definition, 1-7
  priority select data transfers, 1-14
AND/OR DMA1 bit, 4-62
AND/OR DMA2 bit, 4-62
ANYINT bit, 4-58
ARB1[3-0] bit, 4-65
ARB2[3-0] bits, 4-71
arbitration. *See* bus arbitration.
architecture of MC-MXI. *See* MC-MXI architecture.
AS* signal
  basic data transfers, 1-12
  block-mode data transfers, 1-13
  definition, 1-7
  indivisible data transfers, 1-13
  priority select data transfers, 1-14

B

Basic MXIbus registers
  MXI Control Register, 4-12
  MXI Device Type Register, 4-9
  MXI ID Register, 4-6 to 4-7
  MXI Logical Address Register, 4-8
  MXI Offset Register, 4-13
  MXI Status Register, 4-10 to 4-11
  register map (table), 4-4
BCD bit, 4-42
BERR bit, 4-47
BERR* signal
  basic data transfers, 1-12
  block-mode data transfers, 1-13

inability of slave to assert DTACK* and BERR* simultaneously (note), 1-12
MXIbus error (BERR) condition, 5-14
priority select data transfers, 1-14
purpose and use, 1-7
bits
A[23-16], 4-46
A[31-24], 4-46
ADSP[1-0], 4-7
AND/OR DMA1, 4-62
AND/OR DMA2, 4-62
ANYINT, 4-58
ARB1[3-0], 4-65
ARB2[3-0], 4-71
BCD, 4-42
BERR, 4-47
BLOCK, 4-49
BRW[1-0], 4-41
BURST1EN, 4-55
BURST2EN, 4-55
CARDEN, 4-66, 5-3
CHCK*, 4-71
CHCKEN, 4-51
CHCKSTAT, 4-71
CLASS[1-0], 4-6
CMDR*, 4-14
COMM[1-0], 4-69
COMMAK, 4-60
COMMEN, 4-56
COMMST, 4-58
definitions, 4-1 to 4-2
DIR, 4-18
DL, 4-47
DL[1-0], 4-19 to 4-20
DMA1EN, 4-55
DMA2EN, 4-55
DMBIRQ, 4-51
DOR, 4-18
ERR*, 4-19
FAIR, 4-71
FAIRMB, 4-50
FHS*, 4-15
FHSACT*, 4-19
HRDRESET, 4-53
INTACK, 4-34
INTRPTR, 4-15
I/O[3-0], 4-68
IOW[2-0], 4-53
LA[7-0], 4-8
LARENRD, 4-30
LARENWR, 4-32
LBARB, 4-65
LOCKED*, 4-19
LOCKMB, 4-50
LOCKMC, 4-33
M[2-0], 4-42
MANID[11-0], 4-7
MASTER*, 4-15
MBREQ, 4-49
MBSC, 4-50
MDL[11-0], 4-9
MDRQEN, 4-52
MEM[3-0], 4-67 to 4-68
MEMACT, 4-10
MEMACTW, 4-12
M/IO*, 4-34
MISCAK, 4-60
MISCEN, 4-57
MISCINT[1-0], 4-70
MISCST, 4-58
MMBLOCKEN, 4-52
MMEN, 4-51
MMERR[1-0], 4-70
MMERRAK, 4-60
MMERREN, 4-57
MMERRST, 4-59
mnemonics key, B-1 to B-6
MQEDET, 4-20
MW[2-0], 4-54
MXIRQ[1-0], 4-69
MXIRQAK, 4-60
MXIRQEN, 4-56
MXIRQST, 4-58
OFF[15-0], 4-13
OWNMB, 4-49
OWNMBIE, 4-51
OWNMBINT, 4-48
PASSED, 4-11
PCREQ, 4-48
PCREQIE, 4-51
PCREQINT, 4-48
PERR, 4-47
POS1026, 4-65
PROT[3-0], 4-15
PROT[9-4], 4-15
RDERR, 4-21
RDIERD, 4-29
RDIEWR, 4-31

RDRDY, 4-19
RDREQ, 4-30
READY, 4-11
RESET, 4-11, 4-12
RESETSRC, 4-10
RESP[1-0], 4-21
RESP[14], 4-18
RMBIRQ, 4-48
RQMEM[3-0], 4-9
SC[1-0], 4-41
SCCLK[7-0], 4-61
SHMEM*, 4-15
SIG[15-0], 4-16 to 4-17
SIGIERD, 4-29
SIGIEWR, 4-31
SIGREG*, 4-14
SIGREQ, 4-30
SM24OFF[3-0], 4-44
SM32OFF[3-0], 4-43 to 4-44
SRSTENRD, 4-29
SRSTENWR, 4-32
SRSTIERD, 4-29
SRSTIEWR, 4-31
SRSTREQ, 4-30
STAT[12-4], 4-10
S/W DMA1 REQUEST, 4-63
S/W DMA2 REQUEST, 4-63
SWAPEN, 4-34
SWAPMB, 4-51
SYSCLKEN, 4-53
SYSCLKINTEN, 4-56
T[15-0]
    PITLOCAL Register, 4-35 to 4-36
    PITREADY Register, 4-37 to 4-38
    PITSC Register, 4-39 to 4-40
TCENDEN, 4-52
TCIE, 4-52
TCINT, 4-48
TIMERAK, 4-60
TIMERTST, 4-60
TO, 4-48
W1AM[4-0], 4-43
W2AM[4-0], 4-45
W3AM[4-0], 4-45
WRERR, 4-21
WRIERD, 4-29
WRIEWR, 4-31
WRRDY, 4-19
WRREQ, 4-30
BLOCK bit, 4-49
Block Data Rate benchmark, 1-16 to 1-18
block-mode data transfers
    definition, 1-13
    master-mode, 6-9
    programming, 5-10 to 5-11
        advantages, 5-7
        DMA controller method, 5-11
        programmed I/O method, 5-11
    slave-mode, 6-12
    steps comprising, 1-13
BREQ* signal
    bus arbitration, 1-9
    definition (table), 1-7
BRW[1-0] bits, 4-41
BURST1EN bit, 4-55
BURST2EN bit, 4-55
bus arbitration. *See also* data transfers.
    description, 1-9
    Micro Channel, 6-11
    theory of operation, 6-6
BUSY* signal
    bus arbitration, 1-9
    definition (table), 1-7
byte ordering
    common byte-ordering schemes (table), 5-19
    data transfer types (table), 1-8
    description, 1-8
byte swapping, 5-19 to 5-20
    as function of data transfer size
        byte swapping disabled (table), 5-20
        byte swapping enabled (table), 5-20
    purpose and use, 6-4

C

cables, 1-5 to 1-6
    multi-drop cable assembly (illustration), 1-5
    types of cables, 1-5
capability codes
    Micro Channel, A-1
    MXIbus, A-1
CARDEN bit, 4-66, 5-3
CHCK* bit, 4-71
CHCKEN bit, 4-51
CHCKSTAT bit, 4-71

CLASS[1-0] bit, 4-6
CMDR* bit, 4-14
COMM interrupts
  conditions causing, 5-14
  logic for recognizing (illustration), 5-15
  recognizing, 5-15
COMM[1-0] bits, 4-69
COMMAK bit, 4-60
COMMEN bit, 4-56
COMMST bit, 4-58
Communication Control Register, 4-31 to 4-32
Communication Status Register, 4-29 to 4-30
configuration of MC-MXI, 3-1 to 3-2
configuration registers, VXIbus
  Control/Status register, 1-15
  Device Type register, 1-15
  Identification/Logical Address register, 1-15
  Offset register, 1-15
connector. *See* MXIbus connector.
control path, 6-1
control transceivers, 6-5
Control/Status register, VXIbus, 1-15
coupling of dissimilar buses, 5-7
customer communication, *xii*, D-1
cycle termination, master mode, 6-7 to 6-8

D

data path, 6-1
Data Registers, 4-22 to 4-24
  Data Extended (In), 4-22
  Data High (In), 4-23
  Data High (Out), 4-24
  Data Low (In), 4-23
  Data Low (Out), 4-24
data transceivers, 6-5
data transfer types, in byte ordering (table), 1-8
data transfers. *See also* bus arbitration.
  avoiding corruption (caution), 6-8
  basic transfers, 1-12
  block-mode transfers
    definition, 1-13
    programming, 5-7, 5-10 to 5-11
    steps comprising, 1-13
  comparing data transfer rates, 1-16 to 1-18
  indivisible transfers, 1-13
  master-mode, 6-7 to 6-9
    block transfers, 6-9
    cycle termination, 6-7 to 6-8
    indivisible transfers, 6-8
  priority select transfers, 1-14
  slave-mode, 6-11 to 6-12
    block transfers, 6-12
    cycle termination, 6-11 to 6-12
Deadlock (DL) condition, 5-14
deadlocks
  definition, 5-7
  programming considerations, 5-9 to 5-10
device classes, VXIbus, 1-15 to 1-16
device functions, MXIbus, 1-4 to 1-5
Device Type register, VXIbus, 1-15
DIR bit, 4-18
DL bit, 4-47
DL[1-0] bit, 4-19 to 4-20
DMA Control Register, 4-62 to 4-63
DMA Enable Register, 4-55
DMA1EN bit, 4-55
DMA2EN bit, 4-55
DMBIRQ bit, 4-51
documentation
  conventions used in manual, *xii*
  how to use the manual, *xii*
  organization of manual, *xi*
  related documentation, *xii*
DOR bit, 4-18
DS* signal
  basic data transfers, 1-12
  definition, 1-7
  lock-mode data transfers, 1-13
  priority select data transfers, 1-14
DTACK* signal
  basic data transfers, 1-12
  block-mode data transfers, 1-13
  inability of slave to assert DTACK* and BERR* simultaneously (note), 1-12
  priority select data transfers, 1-14
  purpose and use, 1-7

E

electrical specifications, A-2
equipment, optional (table), 2-3
ERR* bit, 4-19
extended register-based or memory-based devices, VXIbus, 1-15

F

FAIR bit, 4-71
FAIRMB bit, 4-50
FHS* bit, 4-15
FHSACT* bit, 4-19

G

GIN* signal, 1-7, 1-9
GND signal, 1-7
GOUT* signal, 1-9

I

Identification/Logical Address register, VXIbus, 1-15
indivisible data transfers, 1-13, 6-8
initializing the MC-MXI, 5-1 to 5-5
  enabling MXIbus System Controller, 5-5
  setting up POS registers, 5-2 to 5-3
  specifying cycle period of System Controller base clock, 5-3 to 5-4
  specifying MC-MXI wait states, 5-3
  timer initialization, 5-4 to 5-5
installation, 3-2 to 3-3
  procedure, 3-3
  unpacking the MC-MXI, 3-1
INTACK bit, 4-34
Interrupt Enable Register, 4-56 to 4-57
interrupts, 5-14 to 5-19
  COMM interrupts, 5-14, 5-15
  generating and handling interrupts, 5-17 to 5-19
    conditions causing interrupts, 6-4 to 6-5
    logic for generating (illustration), 5-18
    steps involving COMM interrupt (example), 5-19
  MISC interrupts, 5-14, 5-16
  MMERR interrupts, 5-14, 5-17
  MXIbus capabilities, 1-14
  MXIRQ interrupts, 5-14, 5-17
  overview, 5-14 to 5-15
  types of interrupts, 5-3
INTRPTR bit, 4-15
I/O space requirements, A-2
I/O[3-0] bits, 4-68
IOW[2-0] bits, 4-53
IRQ* signal, 1-7

K

kit contents (table), 2-3

L

LA[7-0] bit, 4-8
LabVIEW software, 2-4
LabWindows software, 2-5
LARENRD bit, 4-30
LARENWR bits, 4-32
LBARB bit, 4-65
local registers. *See* MC-MXI Local Registers.
local timeout, 5-4
LOCKED* bits, 4-19
LOCKMB bit, 4-50
LOCKMC bit, 4-33

M

M[2-0] bits, 4-42
MANID[11-0] bit, 4-7
manual. *See* documentation.
MASTER* bit, 4-15
Master Mode Page Register, 4-26
master-mode operation, 6-5 to 6-9
  accessing MXIbus memory, 5-7
  address decoder, 6-3
  address formation, 6-4
  address modifiers, 6-4

block-mode data transfers, 5-10 to 5-11, 6-9
byte-swapping logic, 6-4
cycle termination, 6-7 to 6-8
data transfer, 6-7 to 6-9
deadlock, 5-9 to 5-10
definition, 1-4
DOS environment, 5-7
enabling, 6-6
indivisible data transfers, 6-8
interrupt generation and recognition, 6-4 to 6-5
master-mode transfers (caution), 6-5
memory paging, 5-7 to 5-9
MXIbus address broadcast, 6-6 to 6-7
MXIbus arbitration, 6-6
overview, 5-6
parity generator/checker, 6-4
programming considerations, 5-6 to 5-11
situations to avoid, 6-5
state machine, 6-4
timing incompatibilities, 5-10
timing specifications, A-2
Master-Mode PAGE register, 5-7 to 5-8
MXIbus address formation (table), 5-8
programming example, 5-8 to 5-9
MBREQ bit, 4-49
MBSC bit, 4-50
MC INT Ack. Register, 4-60
MC INT Status Register, 4-58 to 4-59
MC-MXI architecture, 6-1 to 6-5
address, data, and control transceivers, 6-5
byte-swapping logic, 6-4
illustration, 6-2
interrupt generation and recognition, 6-4 to 6-5
master-mode
address decoder, 6-3
address formation, 6-4
address modifiers, 6-4
state machine, 6-4
MXIbus terminators, 6-3
overview, 6-1
parity generator/checker, 6-4
slave-mode
address decoder, 6-3
address formation, 6-3
state machine, 6-3
VXIbus device support, 6-5
MC-MXI interface board
illustration, 2-1
kit contents (table), 2-3
Micro Channel features supported, 2-2
MXIbus and VXIbus features supported, 2-2
optional equipment (table), 2-3
optional software, 2-4 to 2-5
overview, 2-1 to 2-2
relationship with MXIbus and VXIbus, 1-14
MC-MXI Local Registers
DMA Control Register, 4-62 to 4-63
DMA Enable Register, 4-55
Interrupt Enable Register, 4-56 to 4-57
Master Mode Page Register, 4-26
MC INT Ack. Register, 4-60
MC INT Status Register, 4-58 to 4-59
MC-MXI STATUS Register, 4-27 to 4-49
MC-MXI-CTRL Register, 4-50 to 4-52
PITCNTR Register, 4-41 to 4-42, 5-4
PITLOCAL Register, 4-35 to 4-36, 5-5
PITREADY Register, 4-37 to 4-38, 5-5
PITRSC Register, 4-39 to 4-40
PITSC Register, 5-5
register map (table), 4-4
SCCLK Register, 4-61
WAITSTATES Register, 4-53 to 4-54
WIN1AM-SMOFF Register, 4-43 to 4-44
WIN3AM-WIN2AM Register, 4-25
MC-MXI STATUS Register, 4-27 to 4-49
MC-MXI-CTRL Register, 4-50 to 4-52
MDL[11-0] bit, 4-9
MDRQEN bit, 4-52
MEM[3-0] bits, 4-67 to 4-68
MEMACT bit, 4-10
MEMACTW bit, 4-12
memory
accessing MXIbus memory in master mode, 5-7
requirements, A-2
memory paging, 5-7 to 5-9
MXIbus address formation using Master-Mode PAGE register (table), 5-8
programming example, 5-8 to 5-9

purpose and use, 5-7
memory-based devices, VXIbus, 1-15
Message-Based Device Registers
A24 Pointer Register, 4-25 to 4-26
A24 Pointer (High), 4-25
A24 Pointer (Low), 4-26
A32 Pointer Register, 4-27
A32 Pointer (High), 4-27
A32 Pointer (Low), 4-27
Communication Control Register, 4-31 to 4-32
Communication Status Register, 4-29 to 4-30
Data Registers, 4-22 to 4-24
Data Extended (In), 4-22
Data High (In), 4-23
Data High (Out), 4-24
Data Low (In), 4-23
Data Low (Out), 4-24
MXI Protocol Register, 4-14 to 4-15
MXI Response Register, 4-18 to 4-21
MXI Signal Register, 4-16 to 4-17
register map (table), 4-4
Slave Configuration Register, 4-33 to 4-34
Soft Reset Service Register, 4-28
message-based devices, VXIbus, 1-15
M/IO* bit, 4-34
MISC interrupts
conditions causing, 5-14
logic for recognizing (illustration), 5-16
recognizing, 5-16
MISCAK bit, 4-60
MISCEN bit, 4-57
MISCINT[1-0] bits, 4-70
MISCST bit, 4-58
MMBLOCKEN bit, 4-52
MMEN bit, 4-51
MMERR interrupts
conditions causing, 5-14
logic for recognizing (illustration), 5-16
recognizing, 5-17
MMERR[1-0] bits, 4-70
MMERRAK bit, 4-60
MMERREN bit, 4-57
MMERRST bit, 4-59
mnemonics key, B-1 to B-6
MQEDET bit, 4-20
Multisystem Extension Interface Bus. *See* MXIbus.
MW[2-0] bits, 4-54
MXI Control Register, 4-12
MXI Device Type Register, 4-9
MXI ID Register, 4-6 to 4-7
MXI Logical Address Register, 4-8
MXI Offset Register, 4-13
MXI Protocol Register, 4-14 to 4-15
MXI Response Register, 4-18 to 4-21
MXI Signal Register, 4-16 to 4-17
MXI Status Register, 4-10 to 4-11
MXIbus. *See also* theory of operation.
addressing modes and address space, 1-10 to 1-11
A16 space, 1-11
A24 space, 1-11
A32 space, 1-11
address modifier codes (table), 1-10
privileges and modes, 1-11
applications
high-speed shared-memory network (illustration), 1-3
multiple mainframe VXIbus or VMEbus system (illustration), 1-3
PC using MXI to control VXIbus or VMEbus (illustration), 1-2
types of applications, 1-2
bus arbitration, 1-9
byte ordering, 1-8
data transfer types (table), 1-8
cables, 1-5 to 1-6
capabilities and uses, 1-1 to 1-2
capability codes, A-1
data transfers, 1-12 to 1-14
basic transfers, 1-12
block-mode transfers, 1-13
comparing data transfer rates, 1-16 to 1-18
indivisible transfers, 1-13
priority select transfers, 1-14
device functions, 1-4 to 1-5
features, 1-4
interrupts, 1-14
overview, 1-1
performance, 1-16
comparing data transfer rates, 1-16 to 1-18
local performance, 1-18

signals
overview, 1-6
summary (table), 1-7
termination, 1-6
VXIbus compared with, 1-14 to 1-16
configuration registers, 1-15
VXIbus device classes, 1-15 to 1-16
word serial protocol, 1-16
MXIbus address broadcast, 6-6 to 6-7
MXIbus arbitration. *See* bus arbitration.
MXIbus connector, C-1 to C-2
illustration, C-1
signal assignments (table), C-1
signal groupings (table), C-2
MXIbus Error (BERR) condition, 5-14
MXIbus master.
See master-mode operation.
MXIbus slave. *See* slave-mode operation.
MXIbus System Controller. *See* System Controller.
MXIRQ interrupts
conditions causing, 5-14
logic for recognizing (illustration), 5-16
recognizing, 5-17
MXIRQ[1-0] bits, 4-69
MXIRQAK bit, 4-60
MXIRQEN bit, 4-56
MXIRQST bit, 4-58

N

NI-VXI bus interface software (table), 2-4
Non-Empty Signal register condition, 5-14

O

OFF[15-0] bits, 4-13
Offset register, VXIbus, 1-15
operation of MC-MXI. *See* theory of operation.
OWN MXIbus (OWNMB) condition, 5-14
OWNMB bit, 4-49
OWNMBIE bit, 4-51
OWNMBINT bit, 4-48

P

PAGE register. *See* Master-Mode PAGE register.
PAR* signal
basic data transfers, 1-12
block-mode data transfers, 1-13
definition, 1-7
priority select data transfers, 1-14
Parity Error (PERR) condition, 5-14
parity generator/checker, 6-4
PASSED bit, 4-11
PC Request (PCREQ) condition, 5-14
PCREQ bit, 4-48
PCREQIE bits, 4-51
PCREQINT bit, 4-48
performance of MXIbus, 1-16
comparing data transfer rates, 1-16 to 1-18
local performance, 1-18
PERR bit, 4-47
physical specifications, A-2
PITCNTR Register
description, 4-41 to 4-42
initializing timers, 5-4
PITLOCAL Register
description, 4-35 to 4-36
initializing timers, 5-5
PITREADY Register
description, 4-37 to 4-38
initializing timers, 5-5
PITRSC Register, 4-39 to 4-40
PITSC Register, 5-5
point-to-point cable, 1-5
POS Registers
overview, 4-64
POS 0/1 MC-MXI Micro Channel ID Register, 4-64
POS 2 MC Setup/DMA Channel 1 Register, 4-65 to 4-66
POS 3 Memory/IO Base Address Register, 4-67 to 4-68
POS 4 MC Interrupt Map Register, 4-69 to 4-70
POS 5 MC Arbitration/DMA Channel 2 Register, 4-71
programming, 5-2 to 5-3
register map (table), 4-5
POS1026 bit, 4-65

priority select data transfers
bus arbitration and, 1-9
procedure for, 1-14
privileges
A16 address space, 1-11
A24 and A32 address spaces, 1-11
Programmable Option Select Registers. *See* POS Registers.
programming
byte swapping, 5-19 to 5-20
controlling interrupts between MXIbus and Micro Channel, 5-14 to 5-19
COMM interrupts, 5-15
generating and handling interrupts, 5-17 to 5-19
MISC interrupts, 5-16
MMERR interrupts, 5-17
MXIRQ interrupts, 5-17
overview, 5-14 to 5-15
initializing the MC-MXI, 5-1 to 5-5
enabling MXIbus System Controller, 5-5
setting up POS registers, 5-2 to 5-3
specifying cycle period of System Controller base clock, 5-3 to 5-4
specifying MC-MXI wait states, 5-3
timer initialization, 5-4 to 5-5
MXIbus master-mode operation, 5-6 to 5-11
accessing MXIbus memory in master mode, 5-7
block-mode transfers, 5-10 to 5-11
deadlock, 5-9 to 5-10
DOS environment, 5-7
memory paging, 5-7 to 5-9
overview, 5-6
timing incompatibilities, 5-10
MXIbus slave-mode operation, 5-12 to 5-13
pseudo-code instructions (table), 5-1
PROT[3-0] bit, 4-15
PROT[9-4] bits, 4-15
pseudo-code instructions (table), 5-1

R

RDERR bit, 4-21
RDIERD bit, 4-29
RDIEWR bit, 4-31
RDRDY bit, 4-19
RDREQ bit, 4-30
READY bit, 4-11
ready timeout, 5-4, 5-10
register-based devices, VXIbus, 1-15
registers
Basic MXIbus registers
MXI Control Register, 4-12
MXI Device Type Register, 4-9
MXI ID Register, 4-6 to 4-7
MXI Logical Address Register, 4-8
MXI Offset Register, 4-13
MXI Status Register, 4-10 to 4-11
definitions, 4-1 to 4-2
MC-MXI Local Registers
DMA Control Register, 4-62 to 4-63
DMA Enable Register, 4-55
Interrupt Enable Register, 4-56 to 4-57
Master Mode Page Register, 4-26
MC INT Ack. Register, 4-60
MC INT Status Register, 4-58 to 4-59
MC-MXI STATUS Register, 4-27 to 4-49
MC-MXI-CTRL Register, 4-50 to 4-52
PITCNTR Register, 4-41 to 4-42, 5-4
PITLOCAL Register, 4-35 to 4-36, 5-5
PITREADY Register, 4-37 to 4-38, 5-5
PITRSC Register, 4-39 to 4-40
PITSC Register, 5-5
SCCLK Register, 4-61
WAITSTATES Register, 4-53 to 4-54
WIN1AM-SMOFF Register, 4-43 to 4-44
WIN3AM-WIN2AM Register, 4-25
Message-Based Device Registers
A24 Pointer Register, 4-25 to 4-26
A32 Pointer Register, 4-27
Communication Control Register, 4-31 to 4-32
Communication Status Register, 4-29 to 4-30

Data Registers, 4-22 to 4-24
MXI Protocol Register, 4-14 to 4-15
MXI Response Register, 4-18 to 4-21
MXI Signal Register, 4-16 to 4-17
Slave Configuration Register, 4-33 to 4-34
Soft Reset Service Register, 4-28
overview, 4-1
POS Registers
overview, 4-64
POS 0/1 MC-MXI Micro Channel ID Register, 4-64
POS 2 MC Setup/DMA Channel 1 Register, 4-65 to 4-66
POS 3 Memory/IO Base Address Register, 4-67 to 4-68
POS 4 MC Interrupt Map Register, 4-69 to 4-70
POS 5 MC Arbitration/DMA Channel 2 Register, 4-71
register map, 4-2 to 4-5
Release-On-Request (ROR) arbitration scheme, 6-6
RESET bit, 4-11, 4-12
reset circuitry on MC-MXI, 6-13
RESETSRC bit, 4-10
RESP[1-0] bits, 4-21
RESP[14] bit, 4-18
RMBIRQ bit, 4-48
RQMEM[3-0] bit, 4-9

S

SC[1-0] bits, 4-41
SCCLK Register, 4-61
SCCLK[7-0] bits, 4-61
SHMEM* bit, 4-15
SIG[15-0] bit, 4-16 to 4-17
SIGIERD bit, 4-29
SIGIEWR bit, 4-31
signals
mnemonics key, B-1 to B-7
overview, 1-6
summary (table), 1-7
SIGREG* bit, 4-14
SIGREQ bit, 4-30
SIZE* signal
basic data transfers, 1-12
block-mode data transfers, 1-13
definition, 1-7
priority select data transfers, 1-14
Slave Configuration Register, 4-33 to 4-34
slave-mode operation, 6-9 to 6-12
address decoder, 6-3
address formation, 6-3
address mapping, 6-10
block transfers, 6-12
cycle termination, 6-11 to 6-12
data transfer, 6-11
definition, 1-4
enabling, 6-9
inability to assert DTACK* and BERR* simultaneously (note), 1-12
mapping A16 space to the MC-MXI register set, 6-10
Micro Channel bus arbitration, 6-11
Micro Channel memory addressing modes (table), 6-10
overview, 6-9
programming considerations, 5-12 to 5-13
state machine, 6-3
timing specifications, A-2
SM24OFF[3-0] bits, 4-44
SM32OFF[3-0] bits, 4-43 to 4-44
Soft Reset condition, 5-14
Soft Reset Service Register, 4-28
software, optional, 2-4 to 2-5
specifications
capability codes
Micro Channel, A-1
MXIbus, A-1
electrical, A-2
physical, A-2
reliability, A-2
requirements, A-2
timing, A-2
SRSTENRD bit, 4-29
SRSTENWR bits, 4-32
SRSTIERD bit, 4-29
SRSTIEWR bits, 4-31
SRSTREQ bit, 4-30
STAT[12-4] bits, 4-10
state machine
master-mode operation, 6-4
slave-mode operation, 6-3

S/W DMA1 REQUEST bit, 4-63
S/W DMA2 REQUEST bit, 4-63
SWAPEN bit, 4-34
SWAPMB bit, 4-51
SYSCLK, specifying cycle period for, 5-3 to 5-4
SYSCLKEN bit, 4-53
SYSCLKINTEN bit, 4-56
System Controller
  altering timeout period with PITSC timer, 5-4
  definition, 1-5
  enabling, 5-5
  specifying cycle period, 5-3 to 5-4
  timeouts, 5-4

T

T[15-0] bits
  PITLOCAL Register, 4-35 to 4-36
  PITREADY Register, 4-37 to 4-38
  PITSC Register, 4-39 to 4-40
TCENDEN bit, 4-52
TCIE bit, 4-52
TCINT bit, 4-48
technical support, D-1
Terminal Count (TC) condition, 5-14
terminating network
  configuration, 3-1 to 3-2
  description, 6-3
  position of networks (illustration), 3-2
  required for daisy-chained devices, 1-6
TERMPWR signal, 1-7
theory of operation
  master-mode operation, 6-5 to 6-9
    block transfers, 6-9
    cycle termination, 6-7 to 6-8
    data transfer, 6-7
    enabling, 6-6
    indivisible data transfers, 6-8
    MXIbus address broadcast, 6-6 to 6-7
    MXIbus arbitration, 6-6
    situations to avoid, 6-5
  MC-MXI architecture, 6-1 to 6-5
    address, data, and control transceivers, 6-5
    byte-swapping logic, 6-4
    illustration, 6-2
    interrupt generation and recognition, 6-4 to 6-5
    master-mode
      address decoder, 6-3
      address formation, 6-4
      address modifiers, 6-4
      state machine, 6-4
    MXIbus terminators, 6-3
    parity generator/checker, 6-4
    slave-mode
      address decoder, 6-3
      address formation, 6-3
      state machine, 6-3
    VXIbus device support, 6-5
  reset circuitry on MC-MXI, 6-13
  slave-mode operation, 6-9 to 6-12
    address mapping, 6-10
    block transfers, 6-12
    cycle termination, 6-11 to 6-12
    data transfer, 6-11
    enabling, 6-9
    mapping A16 space to the MC-MXI register set, 6-10
    Micro Channel bus arbitration, 6-11
Timeout (TO) condition, 5-14
timeouts
  definition, 5-4
  types of timeouts, 5-4
TIMERAK bit, 4-60
timers
  initializing, 5-4 to 5-5
  recommended count values (table), 5-5
TIMERTST bit, 4-60
timing incompatibilities
  coupling of dissimilar buses, 5-7
  programming considerations, 5-10
timing specifications, A-2
TO bit, 4-48
transceivers, 6-5

U

unpacking the MC-MXI, 3-1

V

VXIbus, compared with MXIbus, 1-14 to 1-16
  configuration registers, 1-15
  VXIbus device classes, 1-15 to 1-16
  word serial protocol, 1-16

W

W1AM[4-0] bits, 4-43
W2AM[4-0] bits, 4-45
W3AM[4-0] bits, 4-45
wait states, specifying, 5-3
WAITSTATES Register, 4-53 to 4-54
WIN1AM-SMOFF Register, 4-43 to 4-44
WIN3AM-WIN2AM Register, 4-25
word serial protocol, VXIbus, 1-16
Word Serial Read Request condition, 5-14
Word Serial Write Request condition, 5-14
WR* signal
  basic data transfers, 1-12
  block-mode data transfers, 1-13
  definition, 1-7
  priority select data transfers, 1-14
WRERR bit, 4-21
WRIERD bit, 4-29
WRIEWR bit, 4-31
WRRDY bit, 4-19
WRREQ bit, 4-30